Contents

Appendix A: Circuit Diagram

Appendix B: Assembly Program Listing

Appendix C: Engineering drawings

Appendix D: Numerical results

Results for section 4.1.3: Angular displacement and motor speed

Results for section 4.2: dependence of viscosity on temperature

Results for section 4.3: accuracy and long term stability

1 Introduction

Viscosity is of paramount importance to lubrication engineers, rheologists, chemical engineers, and many more besides. This report describes the background, development, operation and testing of a portable automatic instrument for the measurement of bulk viscosity.

The requirement for such an instrument arose during oxidation studies of new synthetic flourosilicone lubricants, which were carried out by bubbling oxygen through a 20 cm3 quantity at a temperature of 250C. These lubricants have some advantages over conventional hydrocarbon oils: for example they are better suited to harsh conditions such as the extraterrestrial environment. However their high temperature stability is inferior and the oxidation studies were aimed at developing additives to improve performance in this respect.

The condition of the lubricant was monitored by taking hourly samples and measuring the viscosity using a Ferranti-Shirley cone-on-plate viscometer. As the oxidation progressed over several hours or sometimes days, the measured viscosity increased, finally resulting in the formation of a stiff gel. Unfortunately lack of regular data, for example during the night, often resulted in considerable uncertainty regarding the course of the experiment. An automatic viscometer was thus almost essential, particularly for the longer tests. No such instrument was found to be available commercially and so a suitable device was designed and built.

Briefly, this original mechanism stirred the fluid using a glass rod, indirectly driven via a conventional tension spring by a small motor. The extension of the spring was kept constant by means of an electrical contact. Brushes made connection to a simple electronic control circuit which increased the speed while the contact was broken, and decreased it while closed. Thus the rotation rate oscillated around a certain mean value, which was proportional to the viscosity. (See further details on the theory in section 2.) A timer switched the device on at regular intervals for a few seconds, and an output voltage was produced indicating motor speed. This could be plotted on a chart recorder, to obtain a direct graph of relative viscosity against time.

The viscometer just described satisfied the requirement although the accuracy of the measurements was poor. Several possible improvements to the mechanism were foreseen, and in particular the rough nature of the control electronics was an obvious area for further development. In view of this the current project aimed to develop an improved viscometer mechanism and control system. Calibrations and comparisons were also necessary to verity correct operation.

This report considers the theory of fluid viscosity measurement (section 2). The viscometer mechanism, microprocessor control system and operating program are described (section 3). The results of several experiments that were conducted are presented (section 4). Many ideas for further development have occurred during this work, some of which are discussed (section 5). Detailed circuit schematics, program listing, numerical results and the engineering drawings are shown in full in the appendices.

2: Theory of viscosity measurement

2.1 Types of viscometer

The two main types of viscometer are the tube and rotational instruments [ref. 1], The former observe the rate of flow through tubes due to a known pressure difference. These types are unsuitable for this application, because a suitable viscometer must allow the nest of the experiment to proceed normally in the intervals between measurements, The lubrication fluid is contained in a small lest tube or beaker and therefore any tube-like measurements would be impractical and hard to automate.

The vast majority of rotational viscometers fall into two categories: those where two concentric cylinders rotate relative to one another around a common axis; and those consisting of a cone of large vertical angle (approaching 180 degrees), and plate whose plane is through the apex of the cone. Many variations on this theme are possible, but in all types the test fluid is sheared between the rotating parts. The cone on plate type is again rejected for this application, as it would not be possible to perform oxidation experiments inside the viscosity measurement apparatus.

A concentric cylinder viscometer can easily be formed by regarding a beaker in which the experiments are performed as the outer cylinder, and placing a rotating inner cylinder centrally within it. The suitability and simplicity of this arrangement makes it the ideal choice here. Hence the following theoretical derivations are only concerned with instruments of this type.

2.2 Viscosity

Prior to detailed mathematical consideration, it is necessary to define two variables used in the description of fluid flow: shear stress and shear strain. Stress is measured in units of Pascals (1 Pa = I Nm-2). Consider a point P in a body, surrounded by a plane of area A. The material above, below and to the sides of P exert a resultant force F on the element. As the area is varied the force changes, and the ratio F/A approaches a limit as A tends to zero, known as the traction across the area. This traction has a perpendicular component, the 'normal stress', and a parallel component the 'shear stress' s. Shear strain y is defined as the relative displacement of two layers in the fluid, divided by their separation.

A Newtonian fluid is one in which the ratio of shear stress to the rate of shear strain is constant [ref. l].This parameter is the viscosity n. That is,

$n = s / y.$

The unit of viscosity is the poise. Kinematic viscosity v is often used and is defined as

$v = n / p,$

where p is the density of the fluid. The unit of kinematic viscosity is the Stokes; lubricants are usually specified, by convention, in terms of their kinematic viscosity in centistokes (cst).

A non-Newtonian fluid is one in which the viscosity is not a constant parameter, it depends in some way on the shear rate. The Newtonian model is accurate over a large range for most low molecular weight fluids, including water and many aqueous solutions, liquid metals, organic compounds, and silicones. Other fluids such as suspensions obey various more complex models. A large number of such models have been proposed, for example the Bingham [ref. 2], power law fluid [ref. 3], and Casson [ref. 1] models. This report is only concerned with the measurement of viscosity for Newtonian fluids, although modifications to allow for Non-Newtonian behaviour would not be difficult (See discussion, Section 5).

2.3 Concentric cylinder viscometer.

The formulae derived apply to the measurement of Newtonian fluids, confined between concentric cylinders of infinite length, and neglecting any inertial effects [ref. 1]. The inner and outer cylinders are of radius R1 and R2 respectively, and rotate with a relative angular velocity O. Considering the fluid between the inner cylinder and a radius r; each particle moves with a constant angular velocity, such that the net torque on the fluid is zero. The torque G per unit length on a cylindrical surface at radius r is

$G = 2 \pi R1^2 s1$

where s1 is the shear stress on the inner cylinder, The shear stress at any radius r is

$s = G / 2 \pi r^2$

and in particular, at the outer cylinder is

$s2=G / 2 \text{ pi } R2^2$.

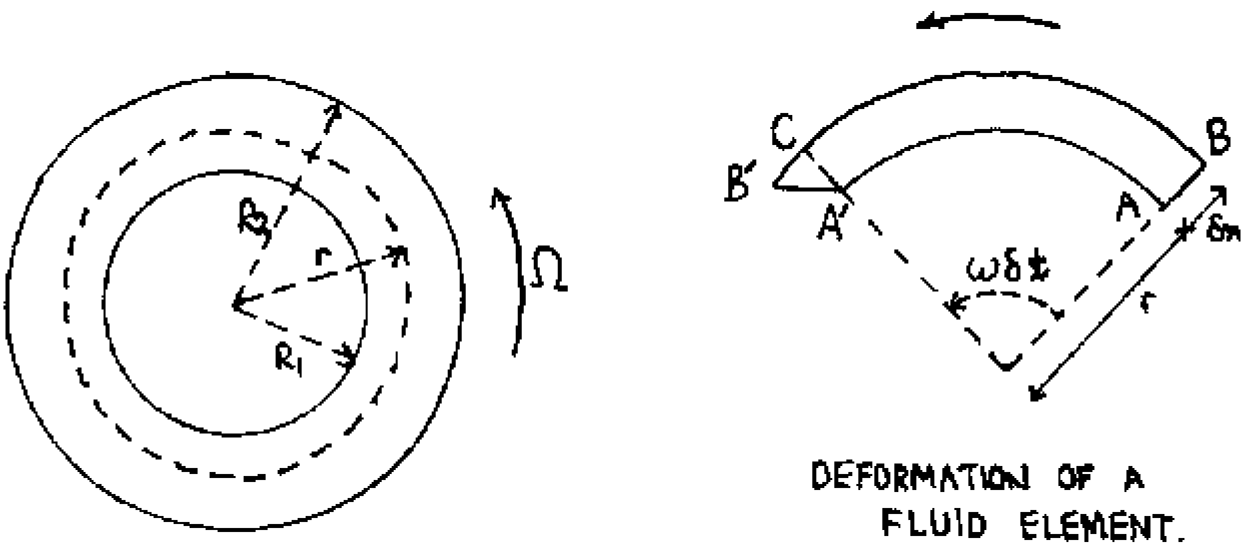


Figure 1: Horizontal section of a concentric cylinder viscometer and deformation of a fluid element.

An expression for the strain rate may be derived using figure 1, which shows sectors of two cylindrical surfaces separated by a small distance dr. In a time dt the radial line AB moves to AB', as opposed to A'C, had the fluid been a rigid body. Now,

$BB' = (r + dr) (w + dw) dt$

and

$BC= (r + dr) w dt$

so the shear strain is

$y = B'C / CA' = (r + dr) dw dt / dr$,

which in the limit as dr tends to zero, gives

$dy / dt = r dw / dr$

Substituting these results into the Newtonian fluid equation leads to

$r dw / dr = G / 2 \text{ pi } v r^2$

Applying the boundary conditions to w = 0 at r = R1, w = O at r = R2, and integrating gives

$O = G (1 / R1^2 - 1 / R2^2) / 4 \text{ pi } n$

For a Newtonian fluid a graph of angular velocity against torque per unit length will be linear through the origin, and have gradient

$(1 / R1^2 - 1 / R2^2) / 4 \text{ pi } n$

This formula will be used to obtain the viscosity. If the instrument is calibrated with a liquid of known viscosity, or used to measure relative viscosity, then all subsequent measurements can be referenced to this and it is not necessary to know R1 or R2 explicitly, provided they remain constant.

2.4 Deviations from the ideal model

2.4.1 End effects

A practical concentric cylinder viscometer must be of finite size, and therefore the top and base of the inner cylinder will also exert a torque. Furthermore, near to the ends, the torque per unit length will be reduced since the velocity gradient is no longer radial. This necessitates complex corrections to the formula derived above; however, the torque is still proportional to angular velocity. Hence provided calibration is performed, the end effects will not cause error.

2.4.2 Temperature dependence

Viscosity is highly dependent on temperature. The relation is often found to approximate

v = A exp (Ev / k T)

over a large temperature range, where v is the kinematic viscosity, k Boltzmann's constant, and T the temperature. The constants A and Ev (known as the activation energy for viscous flow) exhibit a large variation between different fluids. This relation was tested, see section 4.2.

The fluid must he kept at a known and constant temperature throughout the measurement. If the concentric cylirider viscometer is used with very viscous fluids at high shear rates, temperature rise due to shear heating can be troublesome. This effect is neglected here, but is considered further in the discussion, section 5.

2.4.3. Departure from circular flow

In concentric cylinder arrangements, fast moving fluids near to the inner cylinder try to move outwards due to the centripetal force. Such a movement is impossible for the liquid as a whole, so local circulation occurs [ref 4]. These 'Taylor vortices' are only formed above a certain rate of rotation, as in figure 2. This secondary flow is still regular but complex, and the relations derived above no longer apply. At still higher speeds the flow becomes turbulent For Newtonian fluids, the 'Reynolds number' is defined as

Re = O R (R2 - R2) / v

where R is the radius of the moving cylinder, and the other variables are as before.
For inner cylinder rotation, Taylor [ref. 5] found that vortices occurred for

Re > 41.3 (R2 / (R2 - R1)) ^ l/2

At a rotation rate of 300 rpm, and with inner and outer cylinders of radius 1 and 2 cm respectively, the
corresponding minimum kinematic viscosity which may measured is 0.005 cst. The current viscometer will not handle such low viscosities.

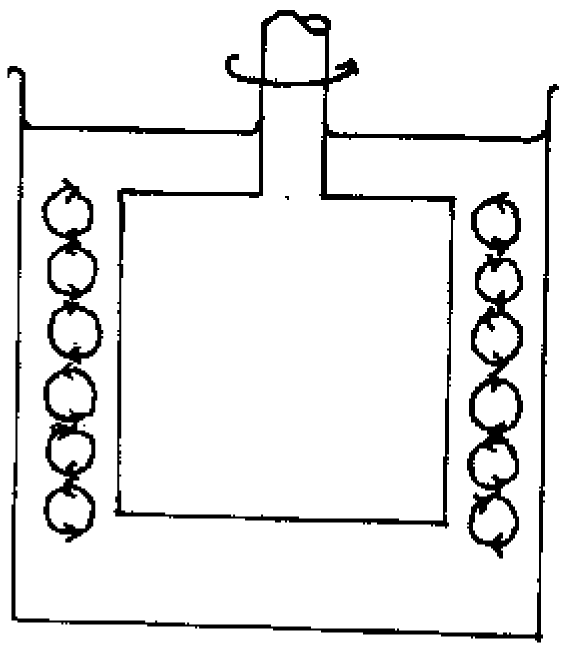


Figure 2: Secondary flow patterns at high rotation rates, known as Taylor vortices.

3.1 Viscometer Mechanism.

Operation of the viscometer mechanism is now described, see figure 3; Complete engineering drawings are shown in Appendix C. All major parts were machined from Dural, with the exception of the stirrer: the corrosive nature of the lubricants under study necessitates the use of glass for this component. In all experiments the assembly was suspended above the test fluid by a retort stand.

A small DC motor (A) drives the disk (E), the outer edge of which is removed over 180 degrees, as shown in the detail. Disk (F) is also cut in a similar way. Disks (E) and (F) may rotate independently about the same axis, due to the bearings in (E) and lower plate (C). The outer end of spiral hair spring (1) is bolted to (E), while the inner end slots into part (F). Therefore the lower disk (F) is indirectly driven by the upper disk (E), via torque spring (I). Connected to (F) by means of a pinned push-fit joint is the chuck (G). Three nylon screws in this component clamp the glass stirring rod (J) securely. The dimensions of the cylindrical, stirring end of (J) restrict the instrument to measurement over a certain range of viscosity. Different ranges can easily be obtained by altering the size of this stirring cylinder. In the following experiments two stirrers were used, one with a cylinder diameter of 2cm and length of 2cm, the other with a cylinder diameter of 1 cm and length 2cm.

An infra-red light beam is emitted by the LED (L), and passes through the mechanism before detection by the photo diode (M). The LED cover (D) ensures that the beam is sufficiently narrow that reflections from parts of the mechanism do not disrupt readings.

The mounting plates (B) and (C) are supported by four corner pillars, (K). Side plates (H) are fitted; these perform the multiple functions of increasing the rigidity of the structure, excluding dirt, and preventing ambient light from corrupting the infra-red beam measurements.

In operation, the motor rotates at a speed up to 300 rpm, determined hy the control electronics (see section 3.2). The stirrer is turned via the spiral spring, which provides a torque proportional to its angular extension. Thus the relative angular displacement between disks (E) and (F) depends directly on the viscosity of the fluid and rotation rate (see section 2.3 theory). As described, parts (E) and (F) are cut away over 180 degrees; this causes the beam to be interrupted once per revolution. Calculation of the mark/space ratio results in the angular displacement, whilst the motor speed may be accurately determined from the period. Figure 4 illustrates the mechanism operation with liquids of different viscosities, and also shows the corresponding expected photo-diode outputs.

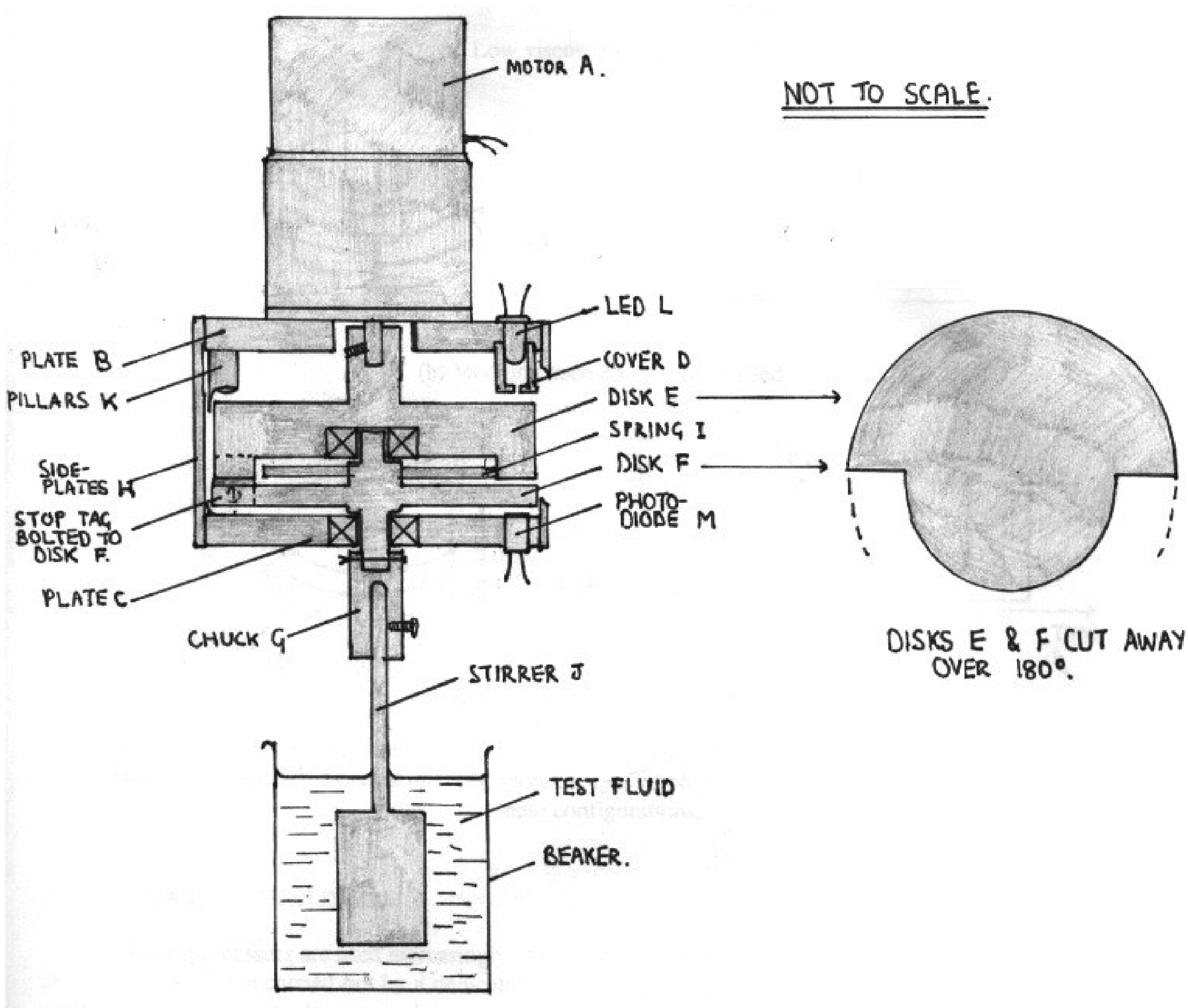


Figure 3: The viscometer mechanism.

A serious problem occurred with the spiral spring. Such a component is difficult to obtain commercially, consequently one was wound from a strip of plate brass approximately 30 cm long, 2 mm wide and 0.5 mm thick. Unfortunately the quality of the spring thus produced was unsatisfactory: only about 3 turns were possible and the spiral shape was difficult to maintain. In addition it was impossible to keep the spring planar: the result of this was that when mounted in the mechanism, the spring scraped on disks (E) and (F) (refer to figure 3). Undoubtedly the instrument's measurement accuracy was seriously affected by the additional friction, hysteresis and non-linearity of this component. Nevertheless useful results were obtainable, as described in section 4, proving the principle and justifying future pursuit of a suitable spring.

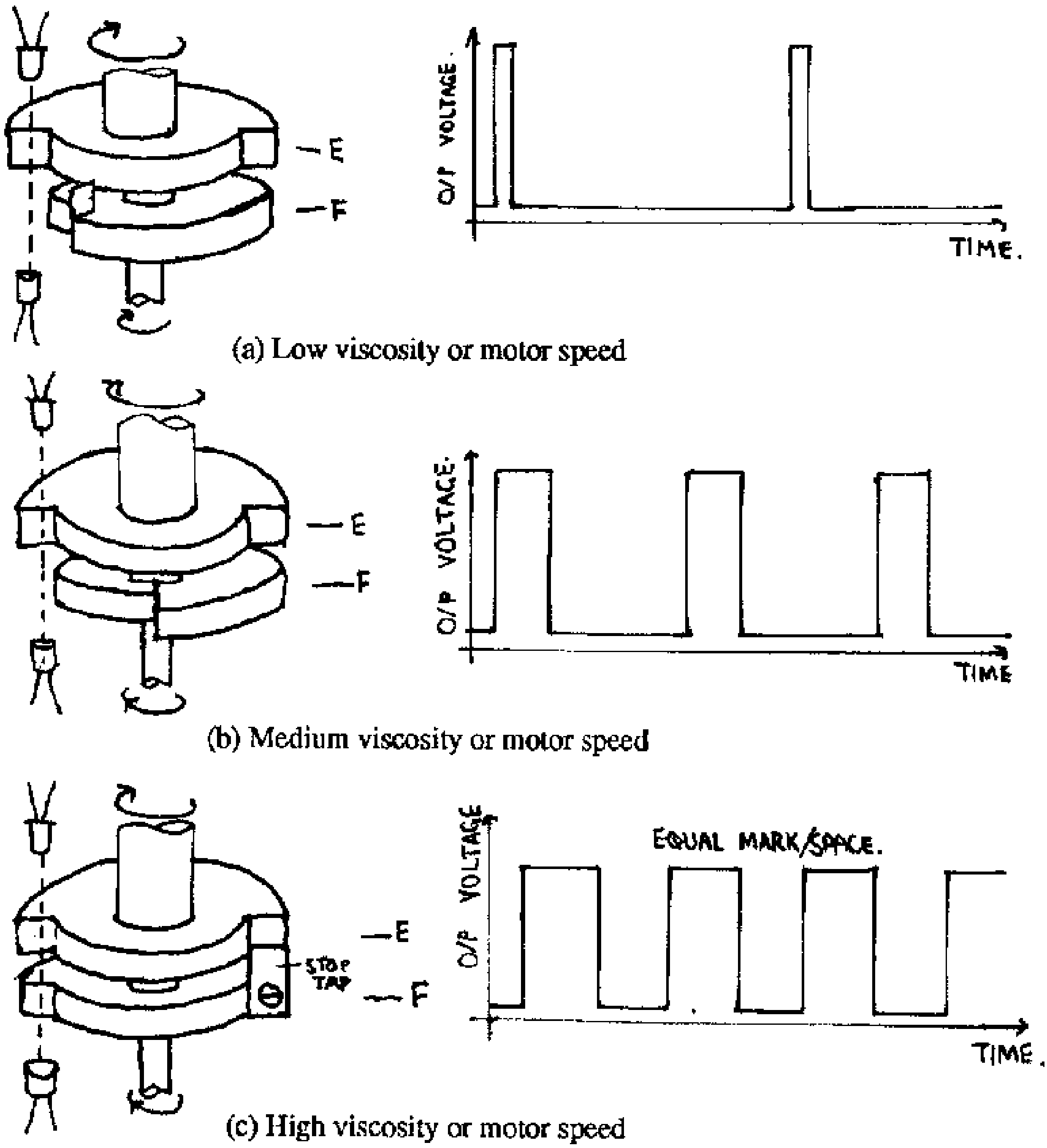


Figure 4: Operation of the viscometer mechanism. The diagrams on the left show the relative orientations of the rotating parts (E) and (F), those on the right show the type of photodiode outputs that may be expected from these configurations.

3.2 Microprocessor control system.

Microprocessors are used increasingly in scientific instruments to handle control functions which would previously have been carried out by a dedicated electronic analogue control circuit A microprocessor control system outperforms its less complex predecessor in almost all respects. It is programmable, so that modifications or specific application requirements may easily be applied without total redesign. The system may be programmed to make detailed calculations and statistical analysis, and adapted to make corrections for non-linearity in the measurement device;

hence greater accuracy is achieved. Data logging can be accomplished with ease, so that operation is totally automatic. Calibration of the instrument also becomes simple, and results can be displayed in a manner superior to that obtainable from an analogue unit. Such advantages justify the additional complexity and cost of a microprocessor based system. For these reasons a microprocessor control unit was designed to control the viscometer mechanism. Of course, a standard commercial computer such as a PC could have been used. However, the expense and inconvenience of dedicating a powerful computer was felt to be excessive, and a self-contained unit was considered more attractive.

A block diagram of the circuit used is shown in figure 5. Appendix A may be consulted for a full schematic.

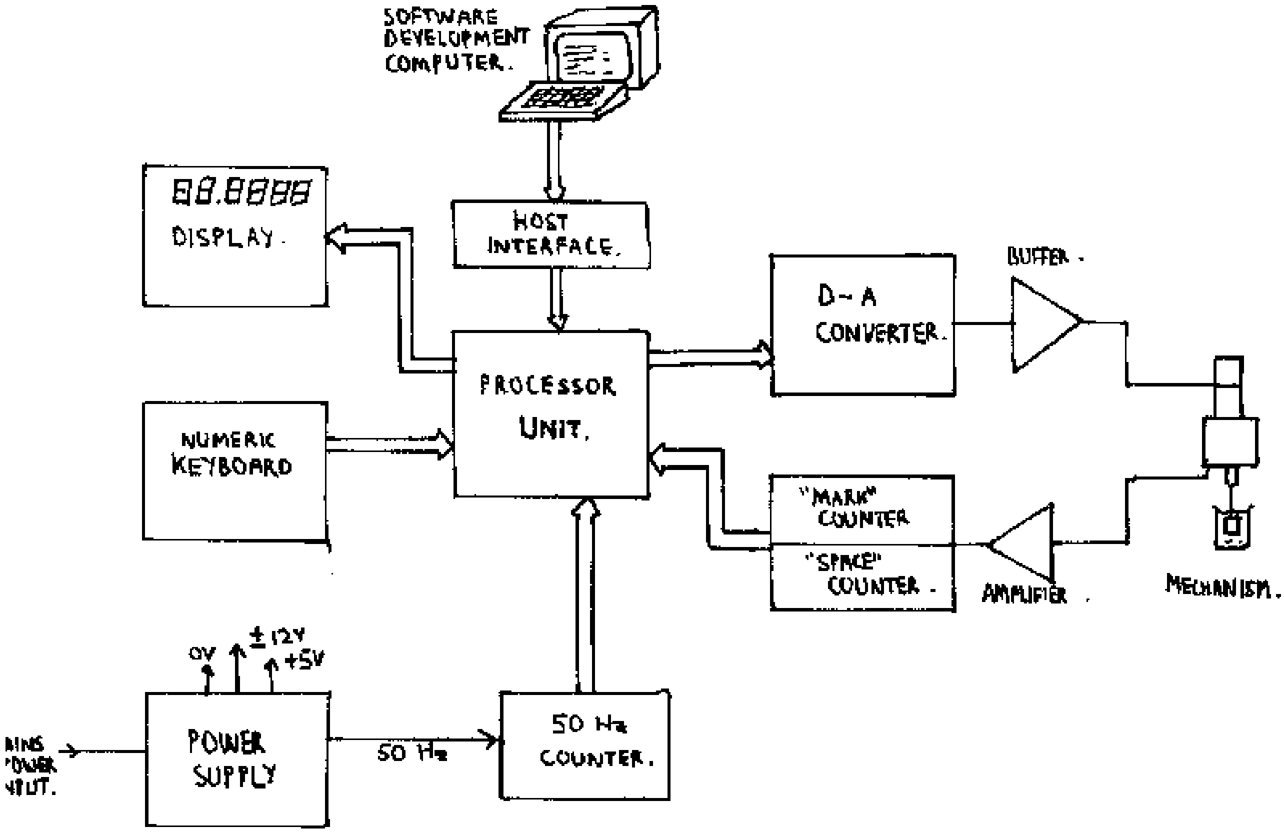


Figure 5: Block diagram of the microprocessor control system.

Processor Unit:

An 8-bit Z80B microprocessor was chosen to form the heart of the system. This processor was certainly the most powerful of its generation and despite today's proliferation of faster 16 and 32-bit processors, the Z80 still has much to commend it. Literature on the device abounds; and in particular, designing a system around the Z80 is considerably simpler than for a more powerful processor, leading to a lower component count and cost. Running at a speed of 3 MHz, the processing power is ample in this application.

The processor unit also includes the necessary logic, 32K of Random Access Memory (RAM) and 8K of Electrically Erasable Programmable Read Only Memory FEPROM). The former is used for variables and data logging, while the latter holds the program itself. EEPROM provides data retention while the system is off, but is electrically erasable as opposed to earlier Ultra-Violet light erasable types. This facilitates reprogramming in situ.

Interface circuit:

This section provides connection between the processor unit and the host computer that was used to develop software; downloading of programs is accomplished without interfering with normal viscometer operation.

Display:

A 6 digit 7-segment LED display and its associated decoder/driver chip provides a digital readout for the system. The display is programmable by the microprocessor and appears as six output ports. A latch has also been incorporated, such that blanking of any or all of the digits is possible: this greatly enhances the clarity of the display by allowing leaning zeros to be removed for example.

Keyboard:

A 20-key numerical membrane keypad, together with decoding logic, allows the user to control the viscometer, enter calibration viscosities, set time intervals, etc.

50Hz counter:

An 8-bit counter is clocked at the 50Hz mains frequency. By reading this input port, the microprocessor can operate a real-time clock and measure time intervals.

Timers:

Two 24-bit counters and input ports time the photo diode illumination and beam broken periods. The counters are clocked at 6 MHz; hence rotation rates down to 21 rpm may be measured, while at the motor's maximum speed of 300 rpm, the periods are accurate to about 1 part in a million (although in practice the precision of the 6 MHz crystal itself limits the accuracy to 30 parts per million).

D-A Converter:

A digital to analogue converter and latch give the microprocessor accurate control over the motor speed. A power operational amplifier buffers the output to a voltage and current sufficient to drive the DC motor. The output voltage is from 0 to +9 volts in 65536 steps, with a differential linearity error of only 0.001%.

Mains power supply unit:

A 12-0-12 volt toroidal transformer, rectifiers and voltage regulators form a standard power supply with +/- 12 V outputs at lA for the D-A converter and buffer. A high efficiency switched mode regulator provides +5V for the digital circuits, which consume a power of approximately 5 W. Use of this unusual type of regulator results in considerably lower overall power consumption and heat dissipation.

3.3 Operating software.

A ZX Spectrum computer running an assembler application was used as a development tool, to edit and compile the operating program prior to downloading into the microprocessor control system's EEPROM. A complete listing of the viscometer control software is presented in appendix B.

A modular programming approach was adopted, and a library of subroutines for elementary functions developed (section 3.3.1). These are called from subroutines which perform statistical analysis, calculate the viscosity, control the period between measurements, and log data (section 3.3.2). These in turn are called from a main control program, under direct user control (section 3.3.3).

3.3.1 Function subroutines:

The collection of subroutines perform the basic functions listed below [ref. 6 & 7]. A 5 byte floating point format is used, consisting of a 32-hit fractional part, 8-hit exponent part, and sign bit. The numbers so represented have a range of +/- 3.4 x 10^38, to a precision of one part in 4.3 x 10^9.

i) Clock/Timer: The 50Hz mains frequency counter is used to drive a real-time clock/timer, which may be displayed in either hours/minutes or hours/minutes/seconds format, according on the user's preference. The 50 Hz counter is 8-bit so in order to retain the correct time, the clock routine must be called at least every 5 seconds. In fact it is called more often, whenever the keyboard is scanned or readings taken, which in practice accounts for most of the processor's time.

ii) Input routines: These scan the keyboard, accept user input of floating point decimal numbers, and convert to the normalised binary floating point format adopted.

iii) Print routine: This allows the program to output a result to the display. The binary floating point number is converted to decimal, and leading zeros blanked.

iv) Error handling: The user is informed of any measurement or arithmetical errors that have occurred. A list of the possible error codes that may be generated is shown in table I below.

v) Arithmetic functions: These operate on numbers in normalised floating point format, and perform the following operations: addition, subtraction, multiplication, reciprocal and division.

vi) Read routines: These read the mark/space ratio and period of rotation, of the viscometer mechanism. If measurement of a viscosity outside the possible range is attempted, an appropriate error is indicated.

vii) Reset: This routine monitors the 'RST' key; if it is pressed for longer than 1/2 a second a system reset is performed. This is similar to switching the machine off and on again, except that none of the results or user defined parameters are erased, with the exception of the time and measurement interval.

viii) Regld, Regsv: These two routines load/save the Z80's registers form/to the stack (upper part of the system memory that is used as temporary storage space by the processor). They are often called at the start and finish of a subroutine, in order that the Z80's registers are preserved by the routine.

i

Code	Cause	Possible solution
E 0	Division by zero	This could occur if a calibration viscosity of zero was entered, or if there was no calibration
E 1	PRINT overflow	The magnitude of the viscosities is too large, use a smaller unit of viscosity
E 2	PRINT negative	*
E 3	Viscosity too large to measure	Fit a smaller glass stirrer, or larger beaker
E 4	Viscosity too small to measure	Fit a larger giass stirrer, or smaller beaker
E 5	Measurement range too small	*
E 6	FPINT conversion error	*
E 7	FPINT negative conversion	*
E 8	Viscosity < Preset minimum	Decrease the pre-set minimum entered
E 9	Viscosity > Preset maximum	Increase the pre-set maximum entered
E A	Too many readings requested	There is only space for 6000 measurements. Enter a smaller number of measurements.

Table I: Explanation of error codes, and possible solutions. *The starred errors should not occur in normal viscometer operation

3.3.2 Viscosity measurement subroutines:

The procedure for viscosity determination is shown in the flow diagram, figure 6. First a useful measurement range is determined, by starting the motor at its maximum rate and gradually reducing the speed in steps until the displacement between the disks in the mechanism falls below 170 degrees. The lower end of the range is found by increasing the motor speed from zero until a displacement is first detected. In order that transitional oscillations in the spring or extreme damping have time to decay, five revolutions of the mechanism are allowed to pass at each speed before reading the angular displacement. Measurements near to the lower end of the range could suffer low resolution and hence inaccuracy; therefore a small amount is added here. Once these two limits have successfully been set, the processor proceeds to take ten readings of displacement angle at ten speeds equally spaced within the range. Again transitional effects are allowed to pass, by now waiting for ten revolutions before reading, at each speed. A statistical analysis [ref. 8] performs a least squares best fit on the data, to find the gradient of the graph of displacement against speed (see section 2.3). This gradient is related to that obtained from a previous calibration reading, and the viscosity calculated.

Having obtained the viscosity, it is then checked to ensure that it lies within a user specified range. If for instance if it becomes to large, no further measurements are taken. This useful function could also be used to sound an alarm if the viscosity of the test fluid deviated too greatly from a specified value. The viscosities are logged in the system memory, where there is space for up to six thousand results. After the experiment, these may be scanned by the operator.

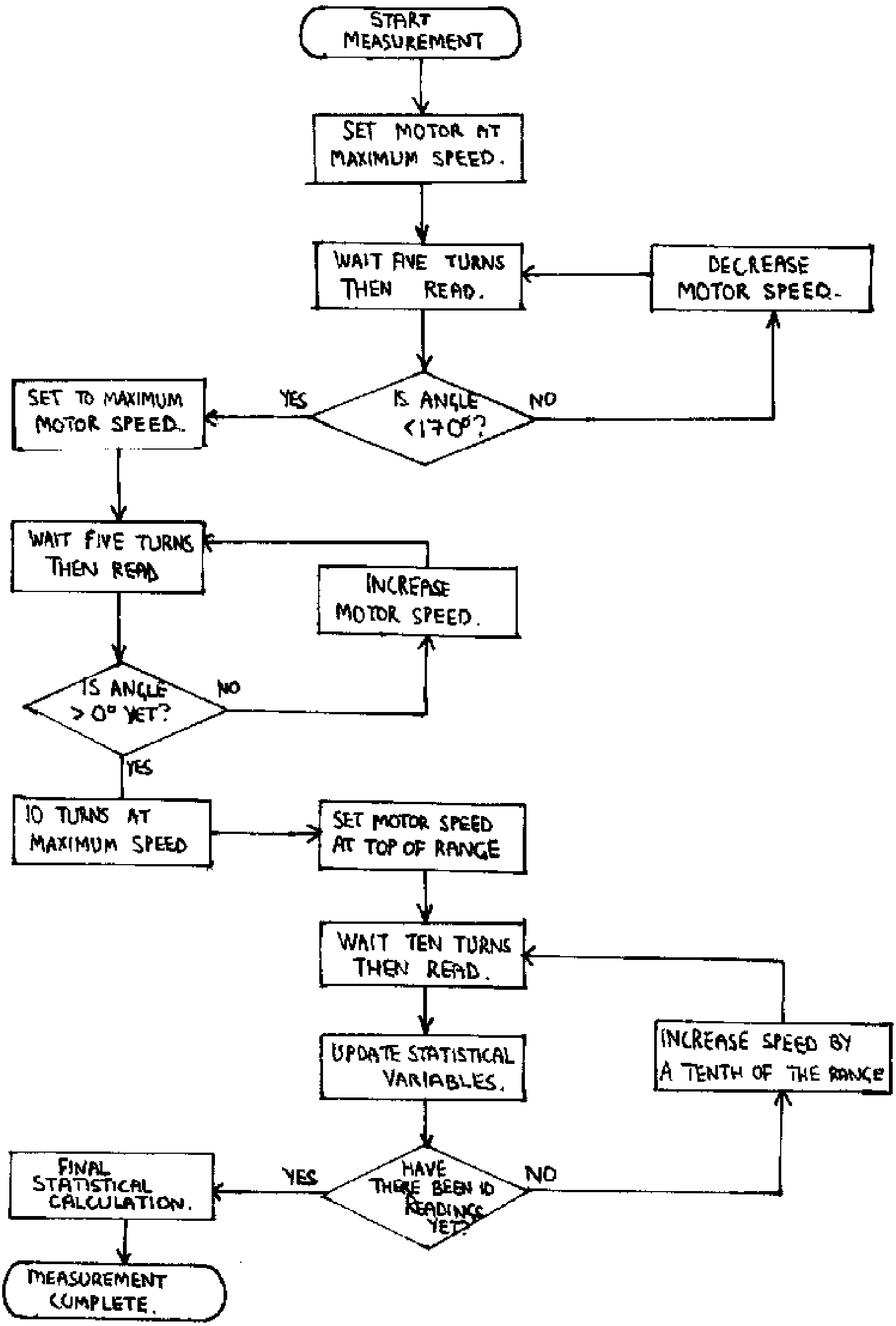


Figure 6: Flow diagram of the subroutine for viscosity measurement.

3.3.3 Main Control Program:

The main control program is under direct control of the user, and calls the various subroutines as required. The user selects the desired operation by typing its code on the keyboard and pressing 'RUN'. A list of the available programs is shown in table 2 below. The system may be reset at any time by pressing the RST' key for longer than 1/2 a second.

The 'A' programs allow the user to enter parameters for the measurements. The time may be set (Al) and the frequency of automatic measurements specified (A2). The desired format of time display is also selectable: type 'A' for hours/minutes format, and 'B' for hours/minutes/seconds format. A calibration reading is carried out by program A3: the user must immerse the stirrer of the mechanism in a fluid of known viscosity, and type in this value (in stokes, centistokes or poise: as long as later measurements are in units consistent with this calibration). The system then performs a set of readings and calculates the gradient of the displacement/speed graph (section 3.3.2). All subsequent measurements are referenced to this value. Minimum and maximum alarm viscosities may be entered (A4), and the number of measurements required specified (A5). Note that programs A3-A5 have no default values, thus it is essential that they are run every time the system is first switched on, although not necessarily following a system reset.

The 'B' programs are for actual viscosity determination. For a oneoff measurement program B1 may be run, while for continuous viscosity monitoring B2 is used. Programs B3 and B4 are for automatic timed measurements, and both log the results in system memory. The interval between measurements, number of readings required, and range are as specified in programs A2, A5, and A4 respectively. The difference between programs B3 and B4 is that the former displays the actual time as set by A3, while the latter displays the time elapsed since the start of the experiment. This would be useful for example in lubricant oxidation experiments.

Program 'C' allows the user to scan through the recorded data, using the +/- keys. The measurement number or viscosity value may be displayed, by pressing the A or B keys respectively.

Program	Function
A1	Set the system clock; default value 0:00
A2	Specify the interval between measurements; default value 1:00
A3	Perform a calibration reading using a fluid of known viscosity
A4	Enter minimum and maximum alarm viscosities
A5	Specify the number of measurements required
B1	One-off viscosity measurement
B2	Continuos viscosity measurement
B3	Automatic operation, with the display showing the clock
B4	Automatic operation, with the display showing the time elapsed
C	Scan through the results of B3 or B4

Table 2: Programs available to the user.

4 Experiments and results

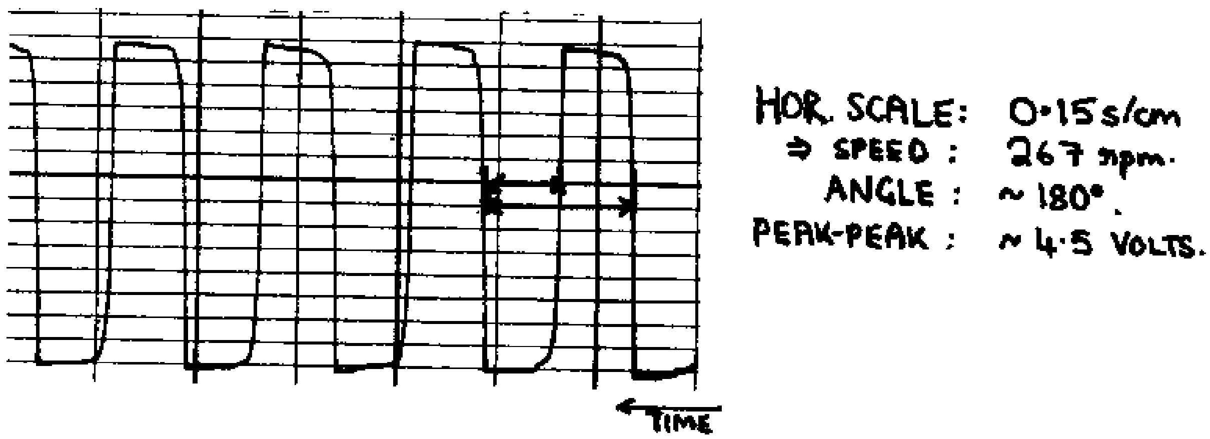
The following experiments aimed to verify the theory (section 2) and correct operation of the viscometer, investigate the dependence of viscosity on temperature, and establish some idea of measurement accuracy.

4.1 Basic viscometer operation

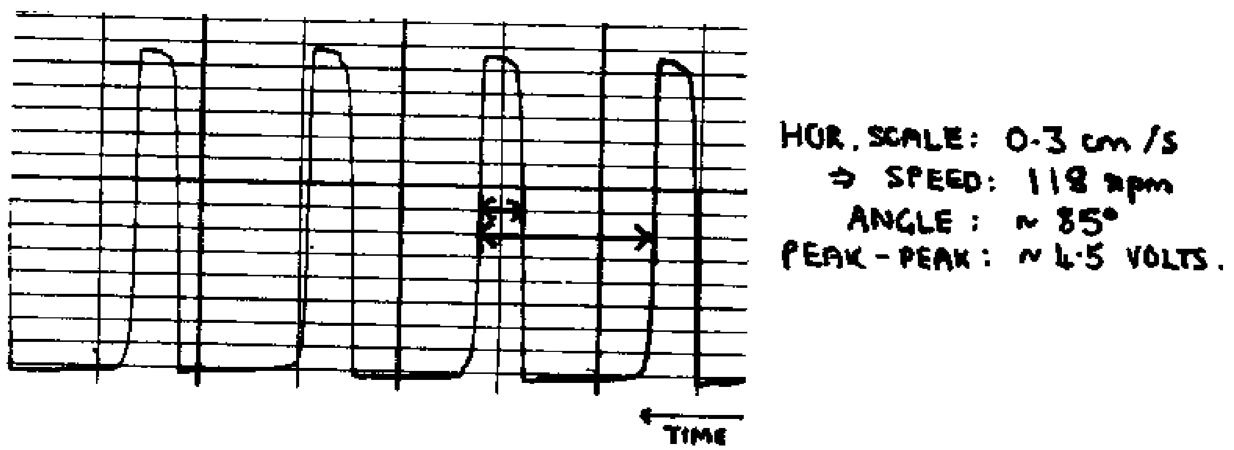
4.1.1 Photo-diode Outout

The output from the photo-diode was monitored using a digital storage oscilloscope, and replayed slowly to a chart recorder, for the same fluid at three different rotation rates. The resulting graphs are shown in figure 7; note that the apparently slow rise and fall time is in fact due to the chant recorder, not the mechanism itself. The top graph shows the output while the motor is rotating at full speed, and the torque generated is sufficient to fully open the spring to an angular displacement of 180 degrees. The middle trace was taken about half way through me measurement when the rotation rate is such that the displacement is roughly 90 degrees. The lower wave form was obtained near the end of the measurement where the motor speed is low. Here there is only just enough torque to extend the spring, and the angle is very small.

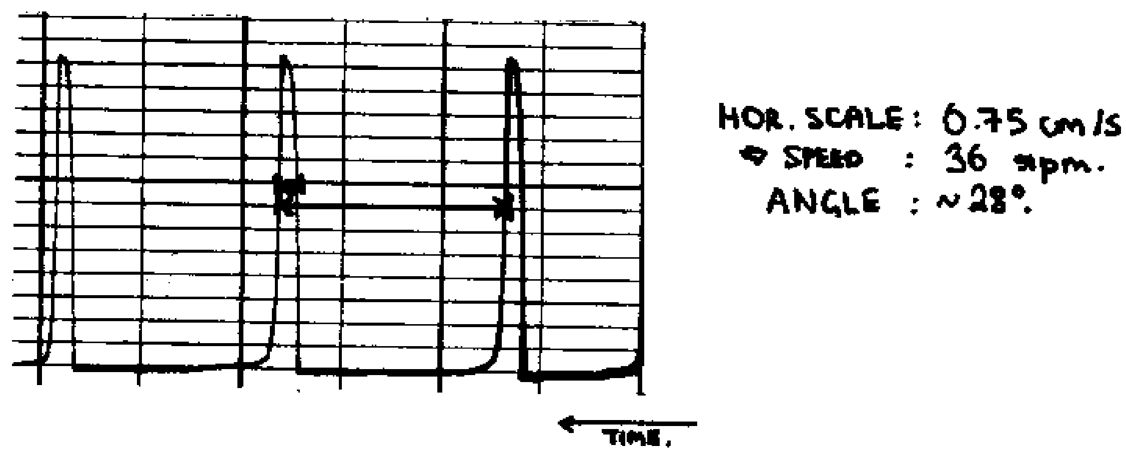
From these graphs it can be seen clearly that the angular displacement of the two disks, indicated by the mark/space ratio of the wave forms, depends on the motor speed. Therefore these results provide some verification of the theory presented in section 2.3 and expectations of section 3.1.



(a): Photo-diode output at high motor speed.



(b) Photo-diode output at a medium motor speed.



(c) Photo-diode output at a low motor speed.

Figure 7: Chart recorder trace of the photodiode output at three different rotation rates. The mark/space ratio of each wave form indicates the displacement angle as a fraction of 360 degrees, while the period is that of the rotation.

4.1.2 Motor Speed during a measurement

In confirmation of the method described in section 3.3.2, figures 8-10 show the variation of motor speed with time during a measurement. The graphs were obtained with three fluids of different viscosity, and using a chart recorder to measure the voltage across the motor. Strictly speaking, due to mechanical losses in the motor, the voltage across it only approximates the speed; however this approximation is sufficient for these largely qualitative graphs. (Note that in all viscosity measurements, the microprocessor control system adjusts the motor speed using the voltage applied, but obtains the exact speed for the calculations from the optical readings.) A measurement takes around 90 seconds, depending on the test fluid's viscosity.

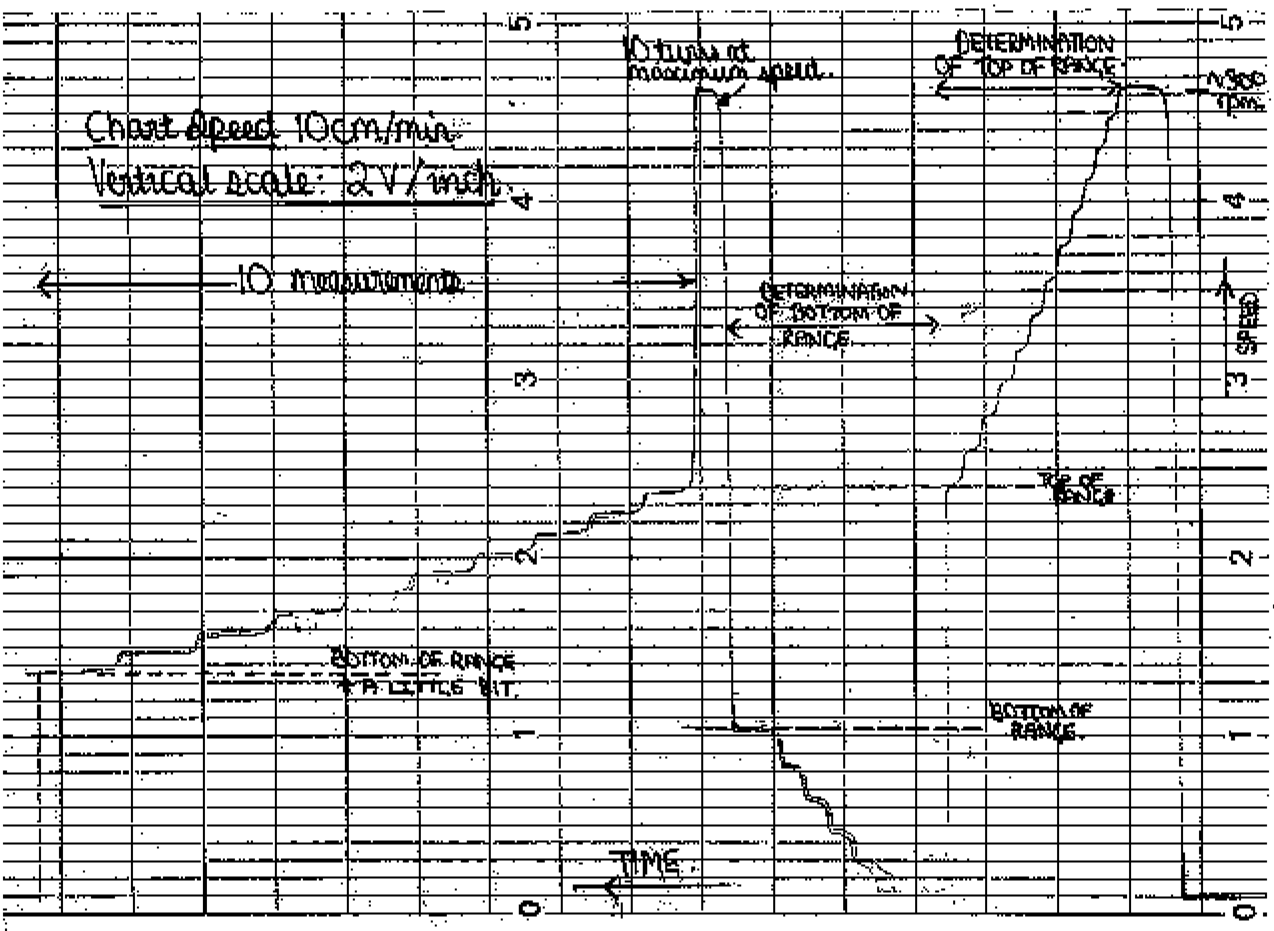


Figure 8: Motor speed during a calibration reading with a 1000 cst fluid

Figure 8 displays the rotation rate for the calibration reading of a 1000 cst fluid. The speed starts off at the maximum (300 rpm) and decreases slowly in steps, until the displacement angle becomes less than 170 degrees. Next the speed is increased from zero until a small angle is detected. Following this ten revolutions at maximum speed are counted, then the measurement run of ten readings equally spaced throughout the range.

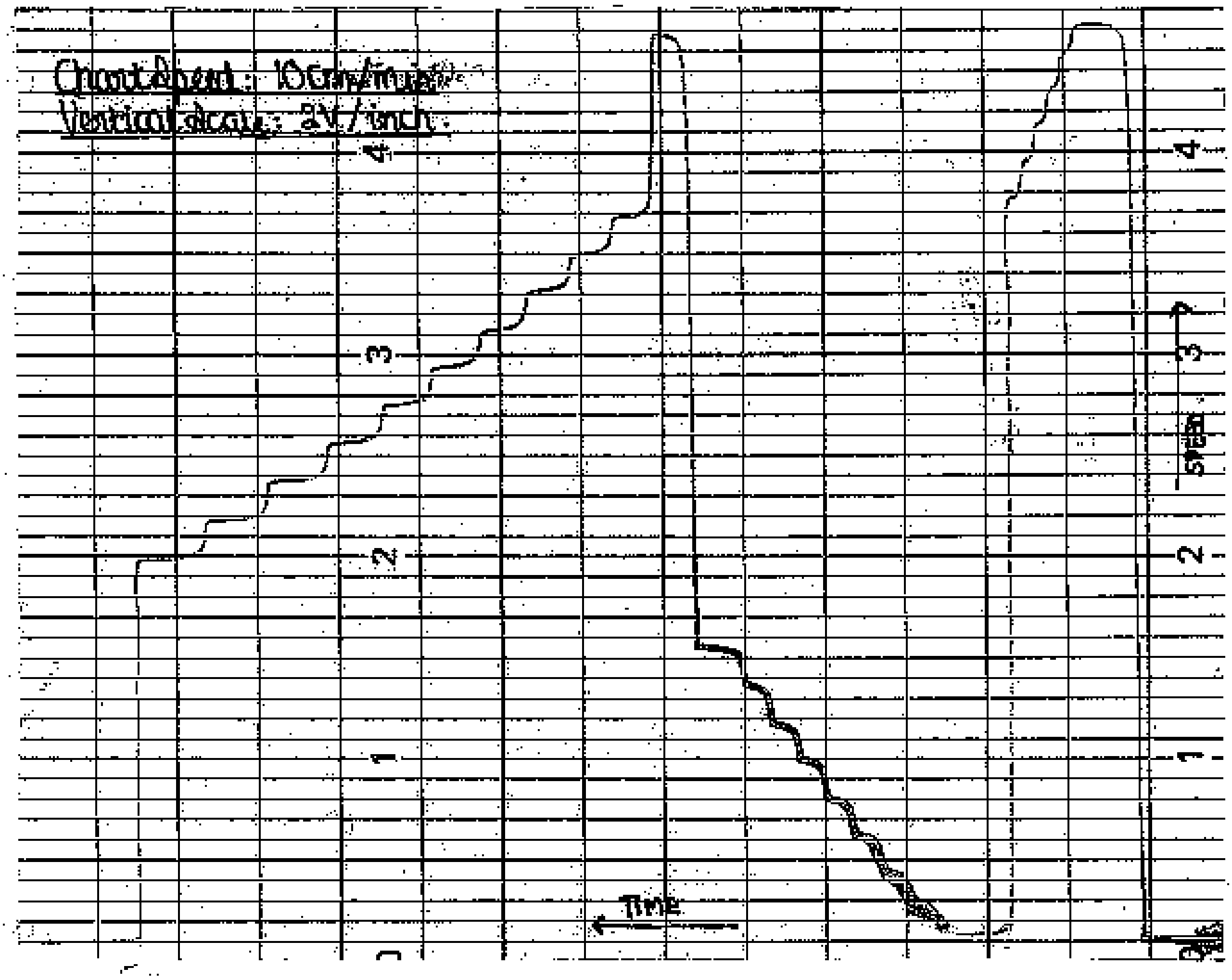


Figure 9: Motor speed during measurement of a 670 cst fluid.

Figure 9 is a similar plot, obtained with a fluid whose viscosity was found to be about 670 cst. The lower viscosity results in the measurement range occurring at a generally higher speed.

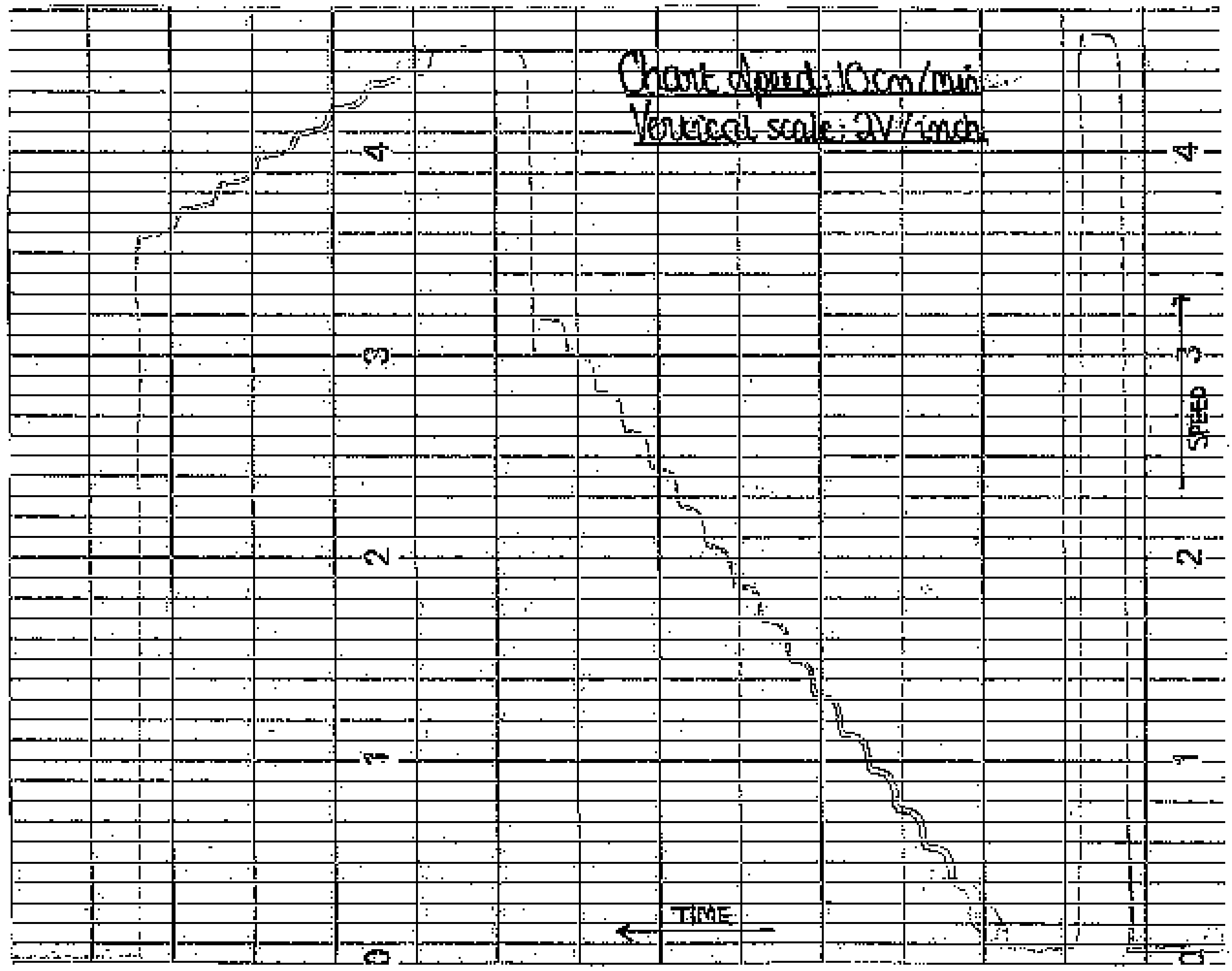


Figure 10: Motor speed during measurement of a 280 cst fluid.

Figure 10 shows the same results for an even lower viscosity liquid, measuring about 280 cst. In this case even at the maximum speed the torque exerted on the glass cylinder was not enough to fully extend the spring to 180 degrees. Nevertheless as long as there is at least some displacement a measurement may still be made, as in this example. Here the fastest motor speed is used as the top end of the range, and a lower limit found in the usual way. (Such matters are transparent to the user.)

4.1.3 Angular displacement and motorspeed.

Figure 11 shows a graph of angular displacement against motor speed, for fluids of approximate viscosity 10000, 1000, and 300 cst, according to the manufacturer's specifications. The results were obtained directly from the control system, before development of the subroutines for calibration and statistical analysis. As expected the relations are linear, with a gradient which increases with viscosity.

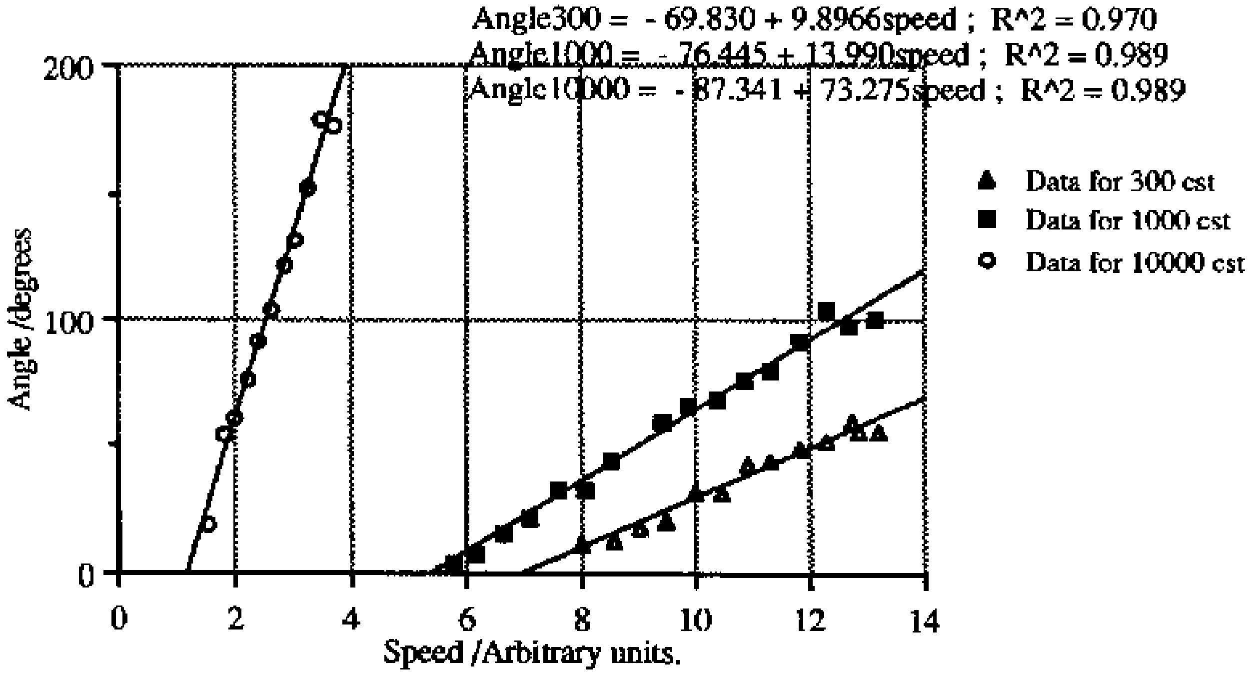


Figure 11; Angular displacement/motor speed for 3 oils of viscosity 300, 1000, and 10000 cst

4.1.4 Conclusions

From the qualitative features of these observations, the viscometer and control system seem to function correctly and as expected. Furthermore the theoretical relations derived in section 2 also appear to be obeyed in this instrument. As expected (see section 3.1) the spiral spring appears to impose some limitations on the measurement accuracy obtainable.

4.2 Dependence of viscosity on temperature

These experiments were intended to test the approximate viscosity-temperature relation mentioned in section 2.4.2. That is,

$$v = A \exp (Ev / k T)$$

where v is the kinematic viscosity, k Boltzmann's constant, T the temperature, and A and Ev constants. The measurements were obtained using both the viscometer of this project (hereafter referred to as the project viscometer), and a Ferranti-Shirley cone-on-plate viscometer. Therefore a useful comparison between the two viscometers was also obtained.

Two lubricants were tested: Dow Corning FS-1265 10000 cst lubricant, and Dow Coming DC200 1000 cst lubricant. These kinematic viscosities are stated by the manufacturer, however the temperature at which the viscosity is specified is not mentioned. However the aim of these experiments was not to determine absolute viscosities, rather to investigate the degree of temperature variation and correlation with an existing commercial viscometer. Hence both instruments were calibrated to read the stated viscosity at room temperature as a reference point.

For the project viscometer measurements, 50 cm3 of the test fluid was contained in a small beaker of diameter 4 cm. This beaker was suspended in a temperature controlled oil bath. The temperature of the test fluid itself was additionally monitored using a digital thermometer, with a thermocouple immersed in the liquid.

During the measurements ample time was allowed for the temperature to reach a stable value, usually slightly less than that of the oil bath due to heat losses from the test fluid. Temperature control is built in to the Ferranti-Shirley viscometer, which pumps heated oil through the instrument to heat the sample under investigation.

Viscosity was measured at temperatures ranging from room temperature to about a hundred degrees centigrade. Above this temperature the oil bath becomes intolerably smelly. Numerical results are presented in Appendix 0.

Figure 12 shows in graphical form the variation with temperature of the 10000 cst fluid. Calibration was performed at 22.5C. The data from the Ferrauti-Shirley instrument approximates the exponential curve extremely well, although that from the project viscometer displays a large spread. This was probably mainly due to the inadequacies of the spiral spring. Error bars plotted on the project viscometer measurements are for a 10% random variation. Temperature is measurable to an accuracy less than half a degree, and the Ferranti-Shirley instrument specifies a precision of under 3%. Thus in the interests of clarity error bars have not been plotted on these data points.

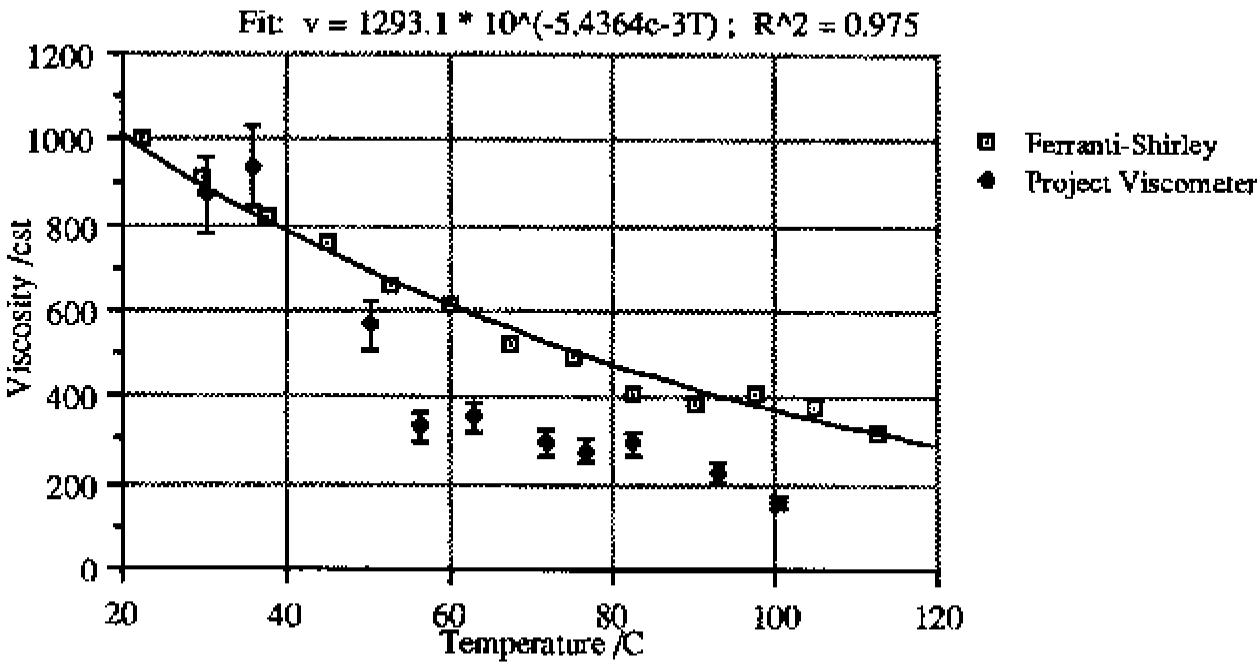


Figure 12: Viscosity/Temperature for the FS-1265 10000 cst fluid

Figure 13 is a graph of viscosity against temperature for the 1000 cst fluid. Again the Ferranti-Shirley results appear more accurate than those of the project viscometer.

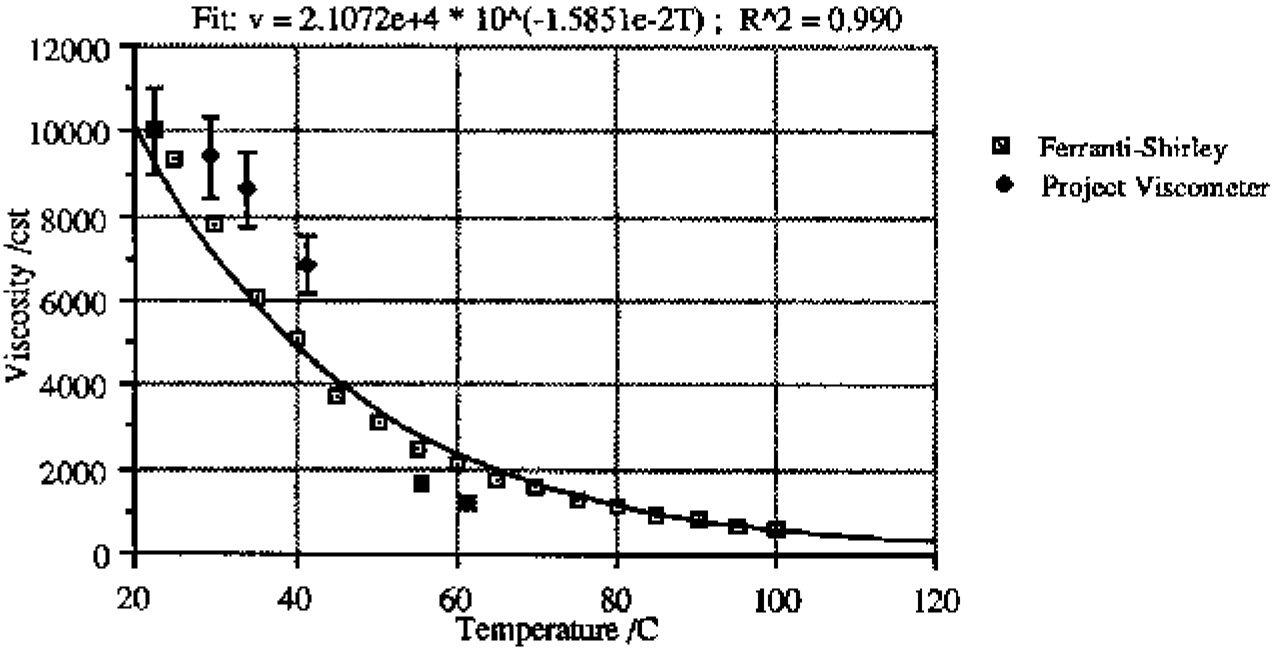


Figure 13: Viscosity / Temperature for the DC200 1000 cst fluid.

These results show that the exponential viscosity-temperature relation is well approximated within the measurement range of the experiment. This is particularly apparent for the 10000 cst FS-1265 lubricant, which displays a rapid decay of viscosity with rising temperature.

4.3 Accuracy and long term stability

In order to investigate the accuracy of the measurements end stability of the mechanism, the instrument was programmed to take 128 readings at half-hourly intervals over a continuous 64 hour period. Dow Corning DC200 1000 cst lubricant was used as the test fluid, and the calibration performed for 1000 cst at 24.1 degrees centigrade. The experiment was carried out at room temperature with no oil bath.

The instrument satisfactorily made all the measurements, a full listing of which appear in appendix D; there was no substantial drift in the instrument between start and finish. A routine statistical analysis of the data revealed a mean value of 948 cst and standard deviation of 37 cst corresponding to 4.0 % of the measured value.

The low mean value clearly demonstrates that the accuracy of all measurements cannot be better than that of the calibration: effectively the precision of the measurements is halved. In this case the calibration seems to have been high, resulting in slightly low figures for all subsequent readings.

It is thought that most of the 4 % random errors arise from the non-linearities of the spring already discussed. However viscosity is also highly affected by variations in temperature, and it could certainly be argued that blame for the observed errors could be apportioned to variations in the laboratory temperature during the day/night cycle. In order to calculate the temperature drift that would cause a +/- 4 % viscosity deviation, the coefficients obtained from the temperature experiments of section 4.2 can be used. The constants from the exponential fit of figure 13 imply that a 37 cst change on 1000 cst would require a temperature change of just 4%. Such an amount may seem excessive but certainly a substantial fraction of the deviations could be due to temperature variations of one or two degrees.

In order to determine the true effect of temperature variations on the figures. future experiments could involve accurate temperature control during the measurements, and/or logging of temperature as well as viscosity.

5 Discussion

Several modifications and enhancements have been foreseen, some of which are discussed here. These further developments would be possible in the mechanism, microprocessor control system, and operating software.

Mechanism:

The mechanism performed well with the exception of the spiral hair spring. Improving this component would undoubtedly result in considerably greater measurement accuracy. A properly machined spring would need to be custom manufactured for this purpose, as none are currently commercially available.

Another problem that was experienced was that the chuck wobbled due to imperfections in the pin-joint between parts (F)and (G) in figure 3; this joint is necessary to enable the mechanism to be assembled. It is thought that a tapped hole in (F) and screw-like end to (G) would result in a screw joint that would substantially reduce this problem.

Mounting the entire mechanism in a rigid frame, and making provision for the test fluid beaker to be held securely beneath it, would eliminate the need for a clumsy retort stand. The experimental conditions would be easier to set up and reproduce accurately.

Microprocessor control system:

As mentioned temperature variations have a large effect on viscosity; therefore some means of temperature control would be advantageous. This function could be incorporated into the existing circuit, by using the digital to analogue converter and an additional comparator to perform analogue to digital conversion of the output voltage from a thermocouple. The processor would extract the temperature from this and compare it with an existing user-specified value. A heater immersed in the fluid and under control of the processor would maintain the liquid at a steady temperature. This modification would greatly enhance the viscometer's usefulness by making the measurements more precise; in addition, temperature-viscosity curves could be obtained automatically by the processor.

Another worthwhile addition would be some form of analogue output, suitable for driving a chart recorder for example. Again the digital to analogue converter could be utilised, followed by a sample-and-hold amplifier so that ordinary motor speed control is not prevented.

A computer interface for PCs and Macintoshes would allow data logging and manipulation if desired. The tedium of copying by hand the 128 results of section 4.3, then typing them all into a calculator, illustrates the obvious advantages of using a computer to handle such mundane tasks. Such an interface could easily be added to the existing circuit, and need not prevent the viscometer's operation as a stand alone unit when desired.

Operating software:

Functional subroutines were developed in an extremely logical manner, which facilitates easy alterations to the operating software. In addition to the modifications required to allow the improvements discussed above, further programs could be incorporated to perform more functions, such as error analysis for example. Compensations for non-linearity or losses in the mechanism can be applied using a polynomial fit to the data, as opposed to the least squares straight line best fit that was used here. This could also be arranged to permit the testing of non-Newtonian fluids (see section 2.2).

Microprocessor control is extremely versatile and many program improvements or application specific requirements may be implemented with ease.

6: Conclusion

The theories of viscosity measurement and temperature dependence considered in section 2 of this report have been verified, and the results additionally used to compare the viscometer's performance against that of a commercially available type, the Ferranti-Shirley cone-on-plac viscometer.

The viscometer developed exceeded expectations in some respects. In particular the versatility of microprocessor control in scientific instruments has become more apparent. The precision of the device was not as high as at fust hoped, although considerably better than that of the original viscometer briefly described in the introduction. It is strongly expected that the blame for this shortcoming lies solely with the inadequate spiral spring, and replacement of this item with a more accurate component should result in immediate and substantial improvements in the precision of the instrument.

Some improvements have been envisaged and the potential of the device demonstrated; it is hoped that these justify further development The instrument should satisfy the requirement for monitoring bulk viscosity in oxidation tests, and with the enhancements suggested, perhaps also be of use in many other applications.

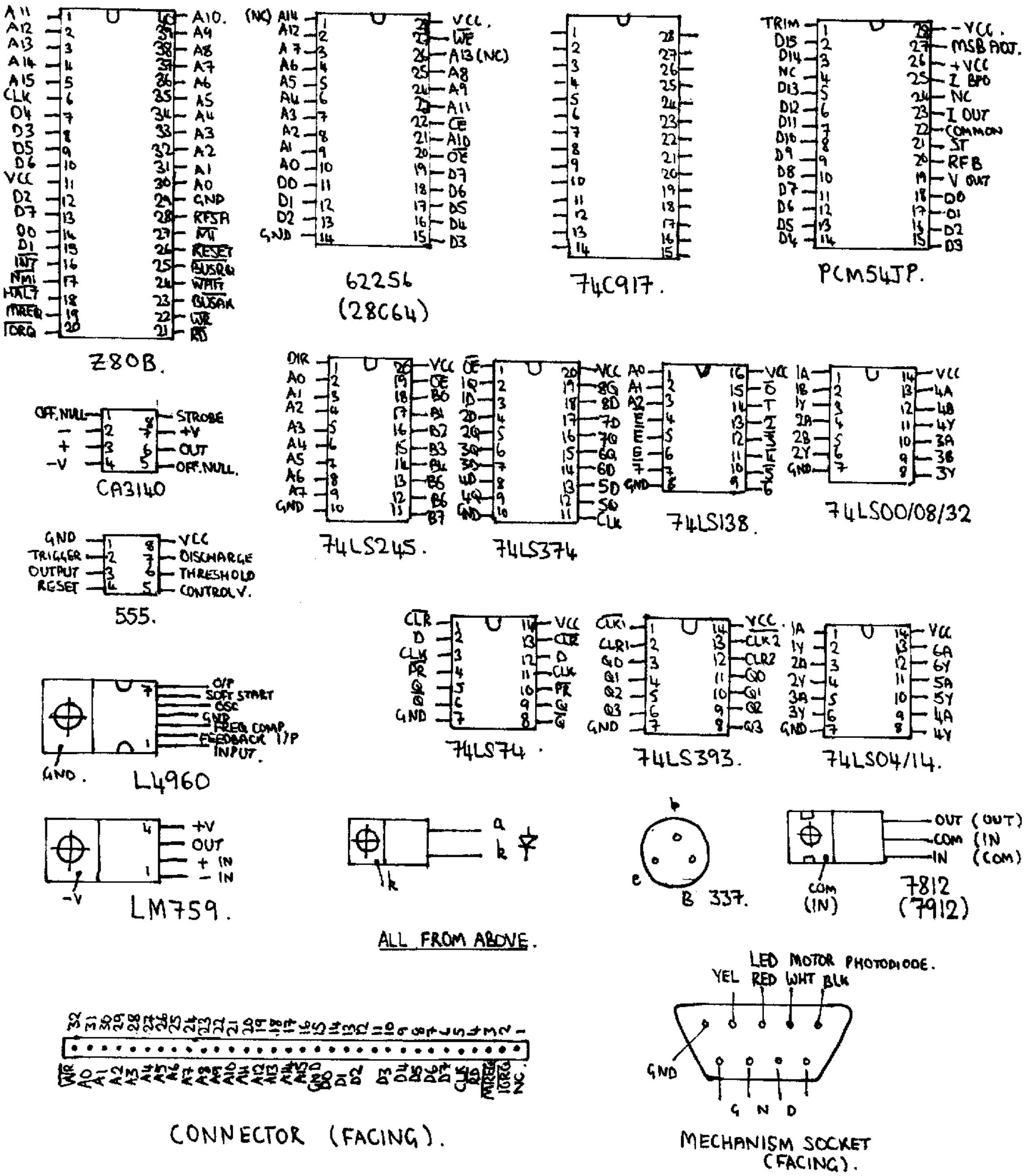
References:

1: Rheological techniques, second edition, by R. W. Whorlow.
2: M. Reiner and R. Riwlin, (1927) Kolloid Z, 43 1.
3: WA. Hyman, (1976) Ind. Eng. Chem. Fundasn. 15215-218.
4: G. I. Taylor, (1923) PIll. Trans. Royal Society A 223 289-293.
5: G. I. Taylor, (1936) Proc. Royal Society A 157 546-578.
6: Z80 & 8080 Assembly Language Programming, by Kathe Spracklen.
7: Practical microcomputer programming: The Z80, by W. 1. Weller.
8: Analysis of straight line data, by Forman 5. Acton.

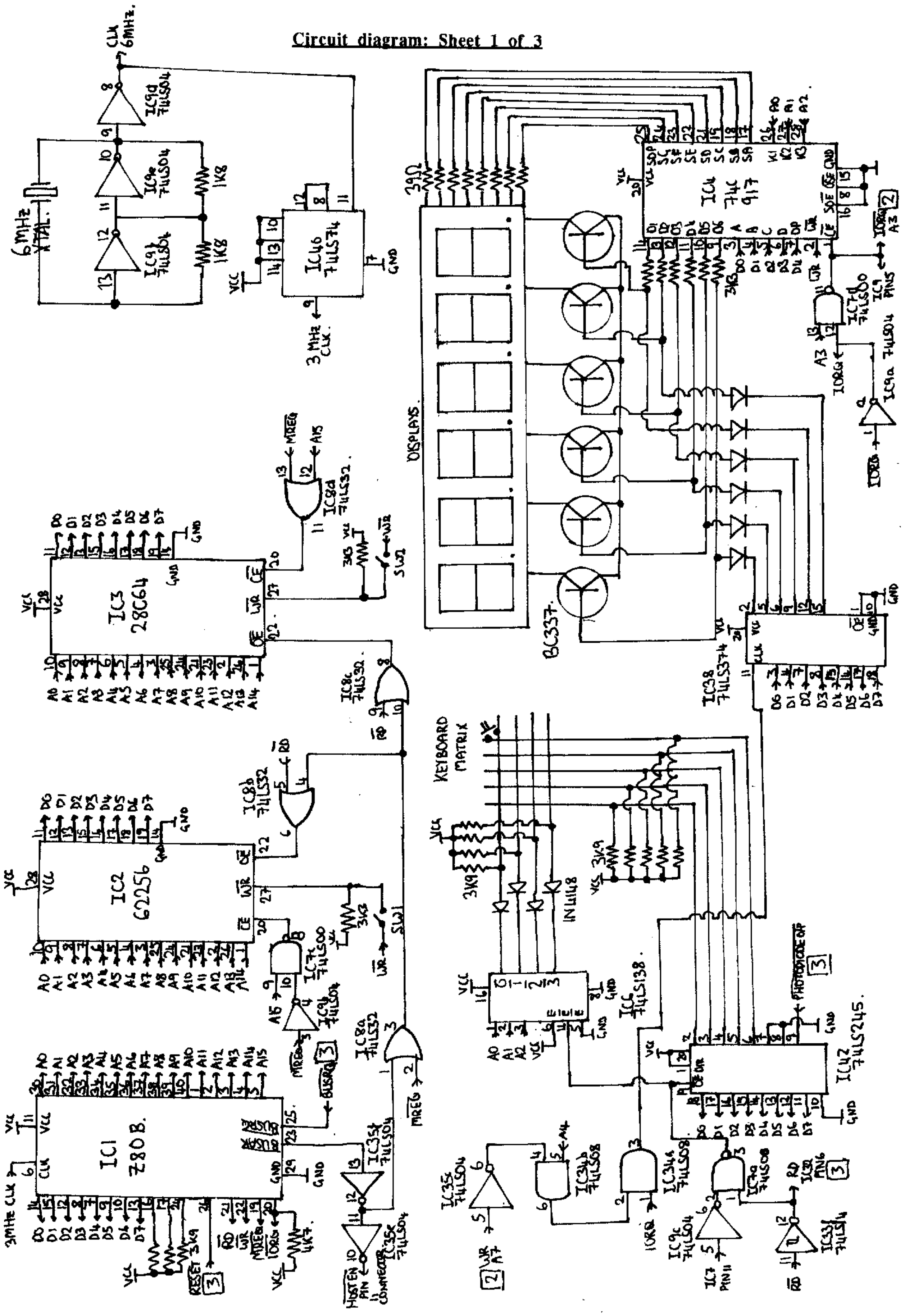
Acknowledgements

Thanks are due to the following people for their help: my supervisors Dr Rochester of the Physics Department and Dr Spikes of the Tribology section, Department of mechanical engineering; Dr Philtipa Cann, Dr Richard Waite, and the other members of the Tribology lab for their advice and comments throughout the project; Jack and Paul the lab technicians, for their assistance with some practical aspects; Chrissy the lab secretary for the administrative details; Martin in the physics workshop for his work in machining the mechanism so precisely and gallant attempts to wind a good spiral spring; and my friend Miss Sheelan Baban for her support and encouragement.

Pinouts for the integrated circuits and allocation of the interface and mechanism connector sockets.

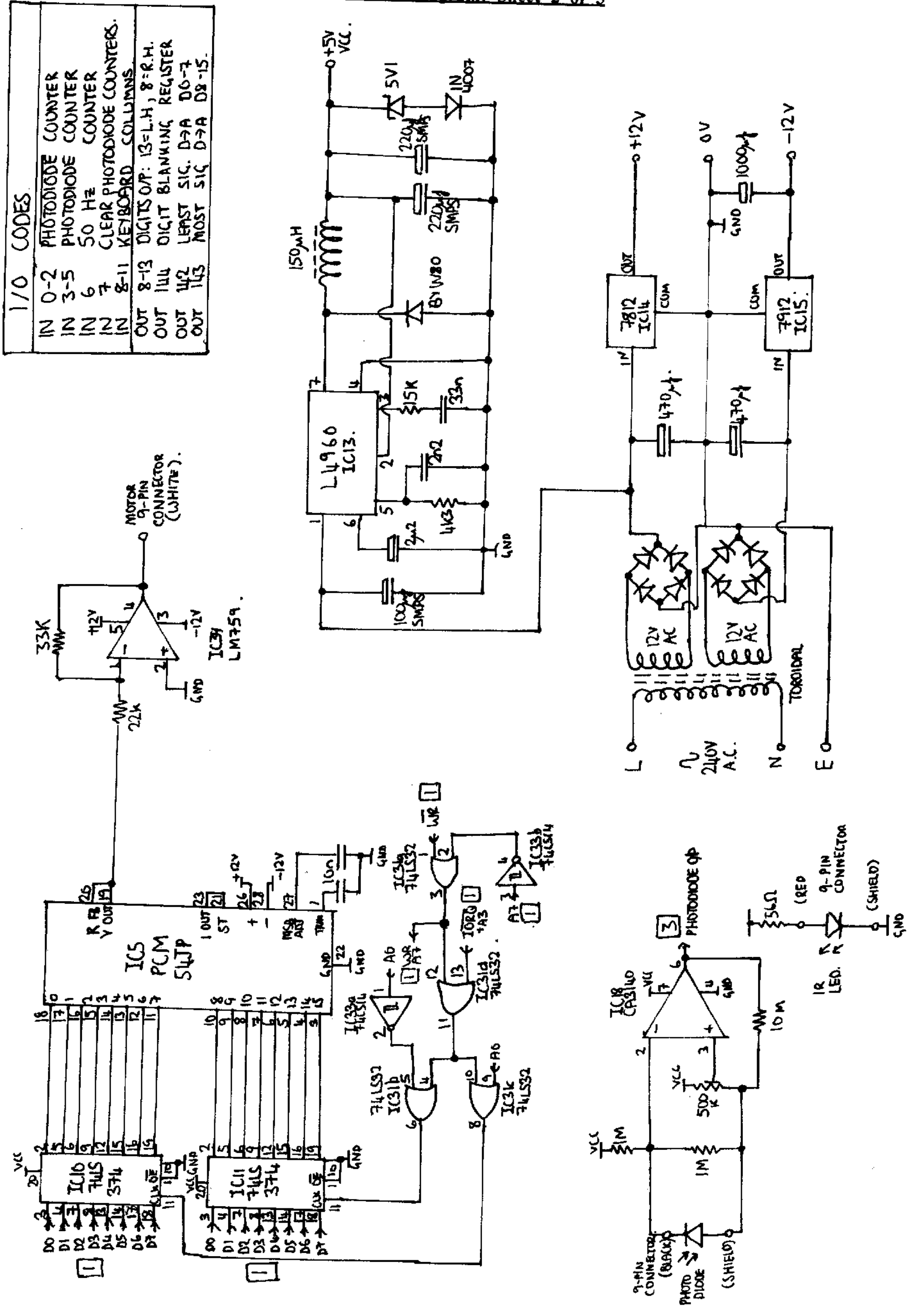


Microprocessor, memory, keyboard and display.



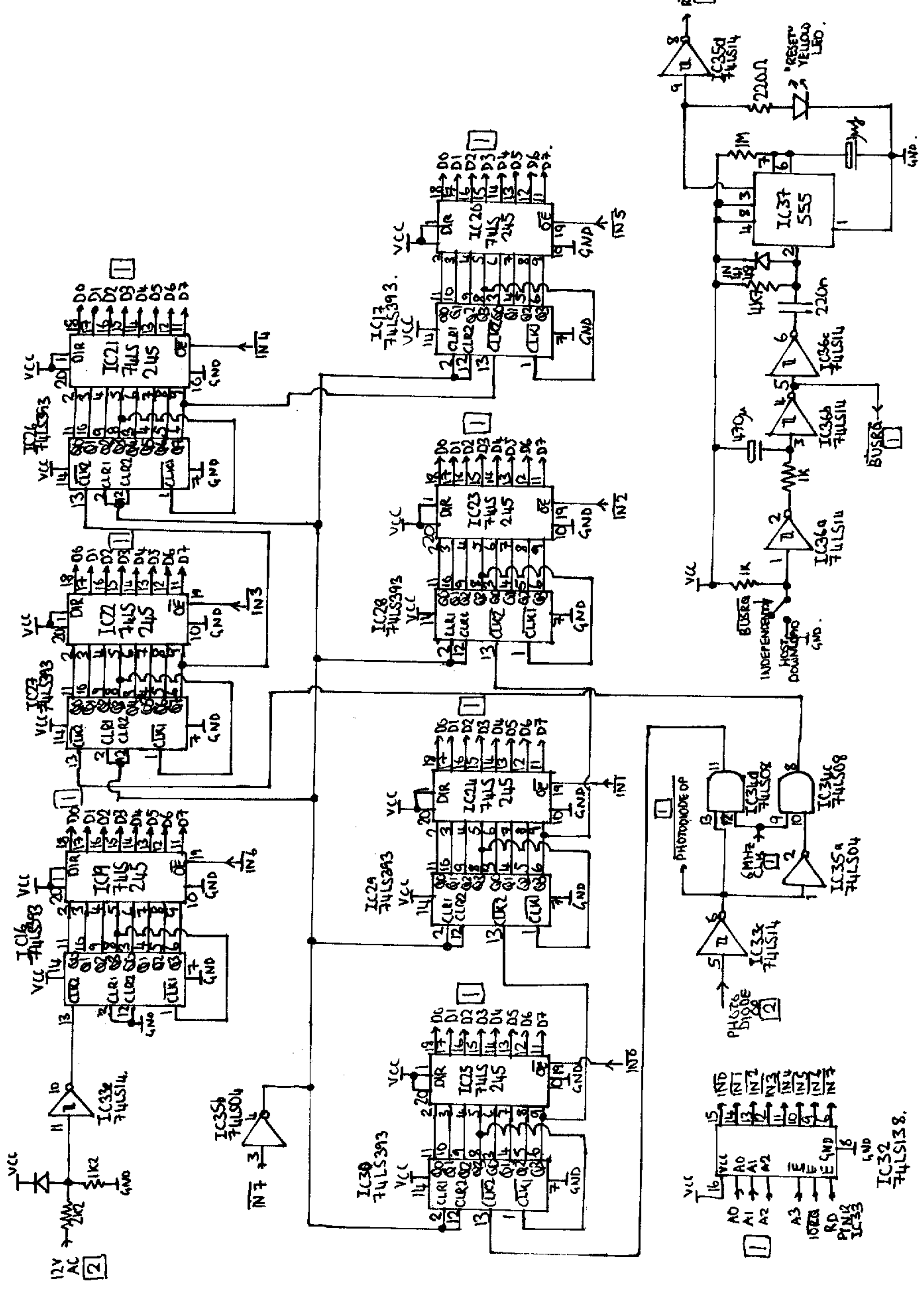
D/A converter, power supply, power amp, photo-diode.

Circuit diagram: Sheet 2 of 3



Timers.

Circuit diagram: Sheet 3 of 3



APPENDIX B: ASSEMBLY PROGRAM LISTING:

Contents:

Program Listing:

Add	Adds two floating point numbers
Alarm	Check to see if the next measurement is due yet
Almset	Get the measurement period from the user
Almvis	Get viscosity measurement range from the user
Calib	Calibration viscosity measurement
Chk1	Finds the maximum motor speed, where angle is < 170 degrees
Chk2	Finds the minimum motor speed, where angle is > 0
Chk3	Checks that maximum motor speed > minmum motor speed
Clkset	User sets the time
Clock	Updates the time and displays it if appropriate flag is set
Divide	Divides two floating point numbers
Equals	Sets a floating point equal to another
Err	Displays the error code
Error	Displays the error code and waits for a keypress
Fpint	Converts a floating point number to a two byte positive integer
Init	Initialises certain necessary variables: called first after a sytstem reset
Input	Get a floating point number from the user
Intfp	Convert a two byte integer to positive floating point format
Keyb	Scan and debounce the keyboard
Main	Main driver routine for all the viscosity programs
Mark	Reads 'Mark' of the mark/space output from the mechanism
Meas	Actual viscosity measurement
Measno	Get number of measurements required from the user
Mlt	Multiply two 32 bit integers
Mult	Multiply two floating point numbers
Norm	Normalise a floating point number
Pb2	Program B2: Continuous viscosity measurement
Pb3	Program B3: Timed viscosity measurements, displaying real time
Pb4	Program B4: Timed viscosity measurements, displaying elapsed time
Pc	Program C: Scan through results
Print	Print a floating point number on the display
Range	Finds minimum and maximum motor speeds for a measurement
Read	Reads mechanism: Angle and motor speed
Recip	Takes the reciprocal of a floating point number
Regld	Load all registers from the stack
Regsv	Save all registers to the stack
Reset	Restarts entire system if the Rst key is pressed for > than 0.5 seconds
Space	Reads 'Space' of the mark/space output from the mechanism
Sub	Subtracts two floating point numbers
Visc	Read absolute, uncalibrated viscosity
Vischk	Check that the measured viscosity is within the measurement range
Wait	Wait for a key to be pressed

E2PROM Program:

The E2PROM is programmed by reversing the positions of IC2 and IC3, and switching both S1 and S2 to 'ON'. The following program is run from RAM, with the subroutines immediately following. To check correct writing, it may be run again with S2 switched to 'OFF'. The location where any error occured will be displayed.

```
STRT    LD      SP,32766
        LD      BC,ROM
        LD      HL,MAINST
; Replace Mainst with Start for part 1.
        SET     7,H
        LD      A,7
        OUT     (144),A
PGM0    LD      A,(BC)
        LD      (HL),A
        LD      E,A
PGM1    LD      A,(HL)
        CP      E
        JR      NZ,PGM1
        INC     BC
        INC     HL
        LD      A,C
        OUT     (11),A
        SRL     A
        RR      A
        RR      A
        RR      A
        OUT     (12),A
        LD      A,B
        OUT     (13),A
        CP      16
        JR      NZ,PGM0
        LD      A,39
        OUT     (144),A
        HALT
; Subroutines follow here.
```

Order of Subroutines:

The actual layout of the Rom is:

```
ROM
        ORG     0
START   LD      SP,65535
        JP      INIT
```

followed by the subroutines in the following order:

Part 1:

DIVIDE
RECIP
SUB
ADD
WAIT
PRINT
ERR
ERROR

INPUT
KEYB
MLT
MULT
NORM
CLOCK
RESET
REGSV
REGLD

Folloed by the constants definitions, followed by:

Part 2:

INIT
MAIN
PB2
PB3
PB4
PC
VISCHK
ALMVIS
MEASNO
CALIB
MEAS
ALARM
ALMSET
CLKSET
VISC
EQUALS
RANGE
FPINT
INTFP
CHK3
CHK1
CHK2
READ
MARK
SPACE

Constants Definition:

```
ZERO    DEFW    0
        DEFW    0
        DEFB    128
TENTH   DEFW    0CCCDH
        DEFW    4CCCH
        DEFB    0FDH
TEN     DEFW    0
        DEFW    2000H
        DEFB    4
TWENTY  DEFW    0
        DEFW    2000H
        DEFB    5
EIGHTY  DEFW    0
        DEFW    2000H
        DEFB    7
ONE70   DEFW    0
        DEFW    2A00H
        DEFB    8
THRE60  DEFW    0
        DEFB    0
```

```
         DEFB    34H
         DEFB    9
KEYS     DEFB    0,1,4,7,16,17,2,5
         DEFB    8,18,19,3,6,9,20
         DEFB    21,12,11,10,22
MTRSPD   DEFB    0D0H
```

Variables Definition:

```
         ORG     32768
LOGN     DEFS    2
LOGNMX   DEFS    2
LOGMAX   EQU     6000
LOGCUR   DEFS    2
LOGFLG   DEFB    1
CLKDOT   DEFS    1
         DEFS    4
RNGMIN   DEFS    5
RNGMAX   DEFS    5
RNG      DEFS    5
RNGINC   DEFS    5
NUM6     DEFS    5
NUM7     DEFS    5
NUM8     DEFS    5
NUM9     DEFS    5
STATXX   DEFS    5
STATXY   DEFS    5
STATYY   DEFS    5
STATX    DEFS    5
STATY    DEFS    5
VISCTY   DEFS    5
NUM5     DEFS    5
TIMERB   DEFS    1
MTRMIN   DEFS    1
MTRMAX   DEFS    1
MLTBUF   DEFS    17
MULTE    DEFS    1
MLTSGN   DEFS    1
NUM1     DEFS    5
NUM2     DEFS    5
NUM3     DEFS    5
NUM4     DEFS    5
CLKSEC   DEFS    2
CLKMIN   DEFS    2
CLKHRS   DEFS    2
CLK50    DEFS    1
CLKFLG   DEFS    1
PRIOVR   DEFS    1
PRIBUF   DEFS    7
PRIEXP   DEFS    1
INPLO    DEFS    2
INPHI    DEFS    2
INPDGT   DEFS    1
INPBUF   DEFS    6
KEYLST   DEFS    1
NXTSEC   DEFS    2
NXTMIN   DEFS    2
NXTHRS   DEFS    2
LMMIN    DEFS    2
LMHRS    DEFS    2
CALVIS   DEFS    2
```

```
VVV     DEFS    5
VISMIN  DEFS    5
VISMAX  DEFS    5
LOG
```

Keycodes:

Key:	Code:
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
A	10
B	11
C	12
RST	16
.	17
-	18
CLR	19
+	20
ENT	21
RUN	22

```
Routine:        Add
Function:       Adds two floating point numbers
Called by:      Read, Sub, Visc
Calls:          Norm
Entry:          BC points to #1,DE to #2, HL points to result
Exit:           Result stored at HL
Preserved:      All
```

```
ADD     CALL    REGSV
        PUSH    HL
        PUSH    BC
        LD      HL,MLTBUF
        LD      B,17
        XOR     A
ADD1    LD      (HL),A
        INC     HL
        DJNZ    ADD1
; Clear Mltbuf for use as temporary storage during addition
        POP     BC
        PUSH    DE
        LD      L,C
        LD      H,B
        LD      DE,MLTBUF+4
        LD      BC,4
```

```
        LDIR
; Transfer first number into Mltbuf for addition.
        LD      A,(HL)
        ADD     A,128
        LD      (MULTE),A
; Multe holds the exponent part.
        POP     DE
        LD      L,E
        LD      H,D
        LD      BC,4
        LD      DE,MLTBUF+12
        LDIR
; Transfer second number into Mltbuf for addition.
        LD      IX,MLTBUF
        JR      Z,ADD2
        SET     0,(IX+16)
ADD2    BIT     7,(IX+15)
        JR      Z,ADD3
        SET     1,(IX+16)
; Test the sign bits and use the last byte of Mltbuf to store thme in. Now these
sign bits are set to 1 as assumed by the floating point format.
ADD3    SET     7,(IX+7)
        SET     7,(IX+15)
        LD      A,(HL)
        ADD     A,128
        LD      B,A
        LD      A,(MULTE)
        SUB     B
        JR      Z,ADD7
; Compare the two exponents to see which is least; if they are equal then the
shifting stage can be skipped.
        JR      NC,ADD4
        LD      IX,MLTBUF
        NEG
        LD      HL,MULTE
        LD      (HL),B
        JR      ADD5
ADD4    LD      IX,MLTBUF+8
ADD5    LD      B,A
; After selecting the appropraite number for exponent equalistaion, shift in
zeros until the exponents become equal. Now the fractional parts may be added.
ADD6    SRL     (IX+7)
        RR      (IX+6)
        RR      (IX+5)
        RR      (IX+4)
        RR      (IX+3)
        RR      (IX+2)
        RR      (IX+1)
        RR      (IX)
        DJNZ    ADD6
ADD7    LD      IX,MLTBUF
        LD      A,(IX+16)
        OR      A
; If both sign bits are positive call Add12 to add the fractional parts.
        CALL    Z,ADD12
        CP      1
; If one or other sign bits was negative then the sign flag holds 1 or 2 and a
subtraction must be performed not an addition. Add16 does this, IX and IY hold
the two numbers in the correct order depending on which sign was negative.
        JR      NZ,ADD8
        LD      IX,MLTBUF
        LD      IY,MLTBUF+8
        CALL    ADD16
ADD8    CP      2
```

```
        JR      NZ,ADD9
        LD      IX,MLTBUF+8
        LD      IY,MLTBUF
        CALL    ADD16
ADD9    CP      3
        JR      NZ,ADD10
; If both sign bits were negative then an addition is needed.
        CALL    Z,ADD12
        SET     7,(IX+7)
ADD10   LD      IX,MLTBUF+4
        POP     HL
        LD      B,4
ADD11   LD      A,(IX)
        LD      (HL),A
        INC     IX
        INC     HL
        DJNZ    ADD11
; Transfer the result to the correct location.
        LD      A,(MULTE)
        SUB     128
        LD      (HL),A
; Store the exponent part at the result location, thus completing the addition
operation.
        CALL    REGLD
        RET
; This subroutine adds the fractional parts and normalises the result. Note that
following addition, the normalisation process is simply a possible 1-bit shift.
ADD12   LD      IX,MLTBUF+3
        LD      IY,MLTBUF+11
        LD      B,5
ADD13   LD      A,(IX)
        ADC     (IY)
        LD      (IX),A
        INC     IX
        INC     IY
        DJNZ    ADD13
        JR      NC,ADD15
        LD      IX,MLTBUF+7
        LD      B,5
; Normalise if necessary by shifting the result and incrementing the exponent.
ADD14   RR      (IX)
        DEC     IX
        DJNZ    ADD14
        LD      HL,MULTE
        INC     (HL)
ADD15   LD      IX,MLTBUF
        RES     7,(IX+7)
        LD      A,9
        RET
; This subroutine subtracts the fractional parts, pointed to by IX and IY.
ADD16   LD      B,8
ADD17   LD      A,(IY)
        SBC     (IY)
        LD      (IX),A
        LD      (IY),A
        INC     IX
        INC     IY
        DJNZ    ADD17
        JR      NC,ADD19
        LD      IX,MLTBUF
        LD      B,8
; In the case of a negative result, complement the fraction to make it positive
and adjust the sign bit accordingly.
ADD18   LD      A,(IX)
```

```
        CPL
        LD      (IX),A
        INC     IX
        DJNZ    ADD18
        LD      IX,MLTBUF-8
; Call subroutine Norm to normalise the result.
        CALL    NORM
        SET     7,(IX+15)
; Set the sign bit indicating a negative result.
        LD      A,9
        RET
ADD19   LD      IX,MLTBUF-8
        CALL    NORM
; Reset the sign bit indicating a positive result.; then call subroutine Norm to
normalise the result.
        RES     7,(IX+15)
        LD      A,9
        RET
```

```
Routine:        Alarm
Function:       Check to see if next
                measurement is due yet
Called by:      pb3
Calls:          None
Entry:          None
Exit:           Nxtsec adjusted if necessary; Z flag set if due
Preserved:      None
```

```
ALARM   LD      IX,NXTSEC
        LD      IY,CLKSEC
        LD      B,6
ALM1    LD      A,(IX)
        CP      (IY)
        RET     NZ
; Compare the alarm time with the current time and return with NZ flag set if
they are not yet equal.
        INC     IX
        INC     IY
        DJNZ    ALM1
; If a measurement is due then compute the next alarm time.
ALM2    LD      IX,NXTSEC
        LD      IY,CLKSEC
        LD      HL,LMMIN
        LD      A,(IY)
        LD      (IX),A
        INC     IX
        INC     IY
        LD      A,(IY)
        LD      (IX),A
; The next alarm time has the same seconds value as the current time. Now the
alarm period minutes and hours (held in Lmmin and Lmhrs respectively) are added
to the current time. Checks are made to ensure that the if the number of minutes
overflows then ten is subtracted and the 10s of minutes variable adjusted
accordingly, and so on for the other numbers.
        INC     IX
        INC     IY
        LD      A,(IY)
        ADD     A,(HL)
        CP      10
        JR      C,ALM3
        SUB     A,10
ALM3    CCF
```

```
        LD      (IX),A
        INC     IX
        INC     IY
        INC     HL
        LD      A,(IY)
        ADC     A,(HL)
        CP      6
        JR      C,ALM4
        SUB     A,6
ALM4    CCF
        LD      (IX),A
        INC     IX
        INC     IY
        INC     HL
        LD      A,(IY)
        ADC     A,(HL)
        LD      C,A
        INC     IY
        INC     HL
        LD      E,0
        LD      A,(IY)
        ADD     A,(HL)
; The hours addition is complicated by the fact that hours are counted in base
24. This is allowed for by computing the new alarm time in one byte, taking
modulus 24, and then extracting the 10s of hours.
        JR      Z,ALM8
        CP      1
        JR      NZ,ALM5
        LD      A,10
        JR      ALM8
ALM5    CP      2
        JR      NZ,ALM6
        LD      A,20
        JR      ALM8
ALM6    CP      3
        JR      NZ,ALM7
        LD      A,30
        JR      ALM8
ALM7    LD      A,40
ALM8    ADD     A,C
        CP      24
        JR      C,ALM9
        SUB     A,24
ALM9    LD      B,0
ALM10   CP      10
        JR      C,ALM11
        SUB     A,10
        INC     B
        JR      ALM10
ALM11   LD      (IX),A
        LD      (IX+1),B
        XOR     A
; The zero flag is set if the measurement period has expired and a reading is
due.
        RET
```

Routine: Almset
Function: Get measurement period from the user.
Called by: Main
Calls: Keyb
Entry: None
Exit: Period is stored in lmmin,lmhrs

Preserved: None

```
ALMSET  XOR     A
        LD      IX,PRIBUF
; Space at Pribuf is used for temporary storage space. The old measurement
period is displayed so that the user can modify it or continue, as required.
        LD      A,(LMHRS+1)
        OUT     (12),A
        LD      A,(LMHRS)
        OUT     (11),A
        LD      A,(LMMIN+1)
        OUT     (9),A
        LD      A,(LMMIN)
        OUT     (8),A
ALMST1  LD      A,(CLKDOT)
; The use of the Clkdot variable permits the current digit to be flashed at 1
Hz, similar to a cursor, using the display blanking register, Out 144.
        OR      A
        LD      A,54
        JR      NZ,ALMST2
        DEC     A
        DEC     A
ALMST2  OUT     (144),A
        CALL    KEYB
        CP      21
        RET     Z
; Return from the subroutine if the Enter key is pressed.
        LD      E,10
        CP      2
        JR      NZ,ALMST3
        LD      E,4
; The E register is used to check the hours digit, which must be less than 4 if
the 10s of hours is 2, less than 10 otherwise.
ALMST3  CP      3
        JR      NC,ALMST1
        OUT     (12),A
        LD      (LMHRS+1),A
; The 10s of hours are stored at Lmhrs+1; subsequent digits are store at Lmhrs,
Lmmin+1 and Lmmin respectively. The following sections use the same general
format for measurement period entry and checking.
        OR      A
        LD      A,54
        JR      NZ,ALMST4
        DEC     A
        DEC     A
ALMST4  LD      (IX),A
ALMST5  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,ALMST6
        SUB     A,4
ALMST6  OUT     (144),A
        CALL    KEYB
        CP      19
; If the Clr key is pressed the subroutine is started again.
        JR      Z,ALMSET
        CP      21
        RET     Z
        CP      E
        JR      NC,ALMST5
        OUT     (11),A
        LD      (LMHRS),A
ALMST7  LD      A,(CLKDOT)
```

```
        OR      A
        LD      A,(IX)
        JR      NZ,ALMST8
        SUB     A,16
ALMST8  OUT     (144),A
        CALL    KEYB
        CP      19
        JR      Z,ALMSET
        CP      21
        RET     Z
        CP      6
        JR      NC,ALMST7
        OUT     (9),A
        LD      (ALMMIN+1),A
ALMST9  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,ALMSTA
        SUB     A,32
ALMSTA  OUT     (144),A
        CALL    KEYB
        CP      19
        JP      Z,ALMSET
        CP      21
        RET     Z
        CP      10
        JR      NC,ALMST9
        OUT     (8),A
        LD      (LMMIN),A
        JP      ALMSET
```

```
Routine:        Almvis
Function:       Get preset alarm viscosities from the user
Called by:      Main
Calls:          Input, Regld, Regsv
Entry:          None
Exit:           Vismin, Vismax hold the preset values obtained
Preserved:      All
```

```
ALMVIS  CALL    REGSV
        LD      HL,VISMIN
        CALL    INPUT
; Get the parameter from the user and store them at Vismin and Vismax. Note that
these have no default values when the system is first switched on.
        LD      HL,VISMAX
        CALL    INPUT
        CALL    REGLD
        RET
```

```
Routine:        Calib
Function:       Calibration viscosity measurement
Called by:      Main
Calls:          Divide, Input, Print, Regld, Regsv, Visc
Entry:          None
Exit:           Calvis holds the calibration ratio
Preserved:      All
```

```
CALIB   CALL    REGSV
```

```
        LD      HL,CALVIS
        CALL    INPUT
; Get the calibration viscosity value from the user.
        CALL    PRINT
        LD      HL,VISCTY
        CALL    VISC
; Call subroutine Visc to measure the raw uncalibrated viscosity.
        RET     C
        LD      BC,CALVIS
        LD      DE,VISCTY
        LD      HL,CALVIS
        CALL DIVIDE
; Calculate the viscosity calibration multiplier.
        CALL REGLD
        RET
```

```
Routine:        Chk1
Function:       Finds the maximum motor speed, where angle > 170.
Called by:      Range
Calls:          Error, Read, Sub
Entry:          None
Exit:           Mtrmax is set; C flag set on error
Preserved:      None
```

```
CHK1    XOR     A
        LD      (MTRMAX),A
        OUT     (142),A
        OUT     (143),A
; Start at the maximum motor speed, ie output zero to the DAC.
        LD      DE,NUM4
        LD      HL,NUM5
        LD      B,10
        CALL    READ
Do ten revolutions initially to get the motor up to speed.
        JR      NC,CHK1C
CHK1A   LD      A,(MTRMAX)
        OUT     (143),A
        LD      DE,NUM4
        LD      HL,NUM5
        LD      B,3
; 3 revolutions per subsequent reading.
        CALL    READ
        JR      NC,CHK1C
; On error jump.
        LD      BC,NUM4
        LD      DE,ONE70
        LD      HL,NUM1
        CALL    SUB
; Subtract 170 degrees to see if the measured angle is < 170.
        LD      IX,NUM1
        BIT     7,(IX+3)
        JR      NZ,CHK1B
; If angle > 170 then decrease the motor speed and try again.
        LD      A,(MTRMAX)
        ADD     A,8
        LD      (MTRMAX),A
        JR      CHK1A
CHK1B   LD      A,255
        OUT     (143),A
; Stop the motor and return.
        OR      A
```

```
        RET
CHK1C   LD      A,255
        OUT     (143),A
; Decide the cause of the error: viscosity too small or too large. Call
subroutine Error to display the error code.
        LD      A,(MTRMAX)
        OR      A
        LD      A,4
        JR      Z,CHK1D
        LD      A,3
CHK1D   CALL    ERROR
        SCF
        RET
```

Routine: Chk2
Function: Finds the minimum motor speed, where angle > 0.
Called by: Range
Calls: Read6
Entry: None
Exit: Mtrmin is set
Preserved: None

```
CHK2    LD      A,(MTRSPD)
        LD      (MTRMIN),A
; Start at the minimum motor speed specified by the constant, Mtrspd. Below this
speed the motor will not turn at all.
        XOR     A
        OUT     (142),A
CHK2A   LD      A,(MTRMIN)
        OUT     (143),A
        LD      B,5
; 5 revolutions per reading.
CHK2B   IN      A,(6)
        LD      (TIMERB),A
; Initialise timer; Read6 is used to check that 2 seconds have not passed. If no
angle is seen in this time, it is assumed that the current speed is too slow,
and a jump is made.
CHK2C   CALL    READ6
        JR      NC,CHK2F
        IN      A,(14)
        RL      A
        JR      C,CHK2C
        IN      A,(6)
        LD      (TIMERB),A
CHK2D   CALL    READ6
        JR      NC,CHK2F
        IN      A,(14)
        RL      A
        JR      NC,CHK2D
        DJNZ    CHK2B
CHK2E   LD      A,255
        OUT     (143),A
; Stop motor and return when a non-zero angle has been measured.
        RET
CHK2F   LD      A,(MTRMIN)
        SUB     A,8
        LD      (MTRMIN),A
; Increase motor speed and try again.
        JR      Z,CHK2E
        JR      CHK2A
```

Routine: Chk3
Function: Checks that minimum motor speed < maximum motor speed
Called by: Range
Calls: Error
Entry: Mtrmin, Mtrmax have been set by Chk1 and Chk2
Exit: C flag is set on error
Preserved: None

```
CHK3    LD      A,(MTRMAX)
        LD      B,A
        LD      A,(MTRMIN)
        SUB     A,8
; Make sure that the difference between the maximum and minimum motor speeds is
positive and greater than 8.
        CP      B
        JR      Z,CHK3A
        JR      C,CHK3A
        RET
CHK3A   LD      A,5
; Signal an error if the range is too small or the minimum speed greater than
the maximum speed.
        CALL    ERROR
        SCF
        RET
```

Routine: Clkset
Function: User sets clock
Called by: Main
Calls: Keyb
Entry: None
Exit: New time is in clksec, clkmin, clkhrs, and 50Hz counter is reset
Preserved: None

```
CLKSET  LD      IX,PRIBUF
        LD      IY,CLKSEC
        LD      B,6
        LD      C,8
        XOR     A
CLKST0  OUT     (C),A
        INC     C
        LD      (IX),A
        LD      (IY),A
        INC     IX
        INC     IY
        DJNZ    CLKST0
; Use Pribuf as temporary storage space and initialise all the time digits to
zero, i.e midnight.
        LD      IY,CLKFLG
        LD      (IY),2
CLKST1  LD      A,(CLKDOT)
; Use of the Clkdot variable allows the current digit to be flashed at 1 Hz,
similar to a cursor, using the digit blanking register, out 144.
        OR      A
        LD      A,54
        JR      NZ,CLKST2
        DEC     A
        DEC     A
```

```
CLKST2  OUT     (144),A
        CALL    KEYB
        CP      21
        JP      Z,CLKSTF
; If the Enter key is pressed, jump to the end.
        LD      E,10
        CP      2
        JR      NZ,CLKST3
        LD      E,4
; Use of the E register here allows the Hours to be checked according to the 10s
of hours entered: if 2 is entered then the Hours must be less than 4, otherwise
less than ten. The other digits are entered and checked in a similar way.
CLKST3  CP      3
        JR      NC,CLKST1
        OUT     (12),A
        LD      (CLKHRS+1),A
        OR      A
        LD      A,54
        JR      NZ,CLKST4
        DEC     A
        DEC     A
CLKST4  LD      (IX),A
CLKST5  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,CLKST6
        SUB     A,4
CLKST6  OUT     (144),A
        CALL    KEYB
        CP      19
        JR      Z,CLKST1
; If the Clr key is pressed, move to the 10s of hours digit again.
        CP      21
        JP      Z,CLKSTF
        CP      E
        JR      NC,CLKST5
        OUT     (11),A
        LD      (CLKHRS),A
CLKST7  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,CLKST8
        SUB     A,16
CLKST8  OUT     (144),A
        CALL    KEYB
        CP      19
        JR      Z,CLKST1
        CP      21
        JR      Z,CLKSTF
        CP      6
        JR      NC,CLKST7
        OUT     (9),A
        LD      (CLKMIN+1),A
CLKST9  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,CLKSTA
        SUB     A,32
CLKSTA  OUT     (144),A
        CALL    KEYB
        CP      19
        JP      Z,CLKST1
        CP      21
        JP      Z,CLKSTF
```

```
        CP      10
        JR      NC,CLKST9
        OUT     (8),A
        LD      (CLKMIN),A
        LD      A,10
        OUT     (13),A
        INC     (IX)
CLKSTB  LD      A,(CLKDOT)
        OR      A
        LD      A,(IX)
        JR      NZ,CLKSTC
        DEC     A
CLKSTC  OUT     (144),A
; Get the time display format: an A entered here (default) will put the clock in
Hours . Minutes mode, while B puts it in Hours.Minutes.Seconds mode.
        CALL    KEYB
        CP      19
        JP      Z,CLKST1
        CP      21
        JP      Z,CLKSTF
        CP      10
        JR      NZ,CLKSTD
        OUT     (13),A
        LD      (IY),2
        JR      CLKSTE
CLKSTD  CP      11
        JR      NZ,CLKSTB
        OUT     (13),A
        LD      (IY),1
        JR      CLKSTE
CLKSTE  JP      CLKST1
; Go back to the 10s of hours digit in case any digit needs to be changed by the
user.
CLKSTF  IN      A,(6)
        LD      (CLK50),A
        XOR     A
        LD      (CLKSEC),A
        LD      (CLKSEC+1),A
; Reset the 50 Hz counter and initialise the seconds and tens of seconds to
zero, then return.
        RET
```

Routine: Clock
Function: Updates clock and displays it if the appropriate flag is selected
Called by: Keyb, Read
Calls: Reset
Entry: Clkflg=0 for no display, 129 for Hrs.Mins.Secs format, 130 for hrs mins format.
Exit: None
Preserved: E,IX,IY

```
CLOCK   CALL    RESET
; Call the Reset subroutine to execute a system reset if the Reset key is being
pressed.
        CALL    CLK5
; Check the 50 Hz counter, and update the clock if necessary.
        CP      25
        JR      C,CLK
; If 0.5 seconds have passed since the beginning of the second, clear Clkdot:
this is used later to flash the decimal point between the Hours and Minutes in
```

```
format A.
        XOR     A
        LD      (CLKDOT),A
CLK     LD      A,(CLKFLG)
        BIT     7,A
        RET     Z
; Return if clock display is not required.
        AND     3
        CP      2
        JR      Z,CLK1
; Display in format B, if Clkflg is 130, otherwise branch to Clk1 and display in
format B.
        LD      A,(CLKSEC)
        OUT     (8),A
        LD      A,(CLKSEC)
        OUT     (9),A
        LD      A,(CLKMIN)
        OR      16
; Or 16 illuminates the decimal point of the digit.
        OUT     (10),A
        LD      A,(CLKMIN+1)
        OUT     (11),A
        LD      A,(CLKHRS)
        OR      16
        OUT     (12),A
        LD      A,(CLKHRS+1)
        OUT     (13),A
        OR      A
        LD      A,254
        JR      Z,CLK0
        INC     A
CLK0    OUT     (144),A
        RET
CLK1    LD      A,(CLKMIN)
        OUT     (8),A
        LD      A,(CLKMIN+1)
        OUT     (9),A
        LD      A,(CLKHRS)
        LD      B,A
        LD      A,(CLKDOT)
        OR      B
; Include a decimal point here only in the first half of each second.
        OUT     (11),A
        LD      A,(CLKHRS+1),A
        OUT     (12),A
        OR      A
        LD      A,52
        JR      Z,CLK2
        INC     A
        INC     A
CLK2    OUT     (144),A
        RET
CLK3    LD      HL,CLKSEC
; This part of the subroutine increments the time by one second. Minutes and
hours are also updates if the seconds overflow.
        LD      C,10
        LD      D,6
        LD      B,0
        LD      A,C
        INC     (HL)
        CP      (HL)
        RET     NZ
        LD      (HL),B
        INC     HL
```

```
        INC     (HL)
        LD      A,D
        CP      (HL)
        RET     NZ
        LD      (HL),B
        INC     HL
        INC     (HL)
        LD      A,C
        CP      (HL)
        RET     NZ
        LD      (HL),B
        INC     HL
        INC     (HL)
        LD      A,D
        CP      (HL)
        RET     NZ
        LD      (HL),B
        INC     HL
        INC     (HL)
        LD      A,C
        CP      (HL)
        JR      NZ,CLK4
        LD      (HL),B
        INC     HL
        INC     (HL)
        RET
CLK4    LD      A,4
        CP      (HL)
        RET     NZ
        INC     HL
        LD      A,2
        CP      (HL)
        RET     NZ
        LD      (HL),B
        DEC     HL
        LD      (HL),B
        RET
CLK5    LD      A,(CLK50)
        LD      D,A
        IN      A,(6)
; Read in the 50 Hz counter value and check it against the value calculated for
the next second.
        SUB     D
        JP      P,CLK6
        NEG
CLK6    CP      50
        RET     C
; Return if a second has not passed yet.
        LD      A,50
        ADD     D
        LD      (CLK50),A
; Calculate the time of the next second, and call Clk3 to increment the clock
counters.
        CALL    CLK3
        JR      CLK5
```

Routine: Divide
Function: Divides two floating point numbers
Called by: Calib, Range, Read, Visc.
Calls: Mult, Recip, Regld, Regsv.
Entry: BC points to #1, DE to #2, HL to result #1/#2
Exit: Result stored at HL. Result bytes unchanged on error

```
Preserved:      All


DIVIDE  CALL    REGSV
        PUSH    HL
        LD      HL,PRIBUF
; Use Pribuf as temporary storage space; calculate the division by first
calculating the reciprocal of the quotient then multiplying by the divisor.
        CALL    RECIP
        POP     HL
        LD      DE,PRIBUF
        CALL    MULT
        RET



Routine:        Equals
Function:       Set a floating point number equal to another
Called by:      Pb3, Visc
Calls:          Regld, Regsv
Entry:          DE points to #1, HL to #2.
Exit:           #1=#2
Preserved:      All


EQUALS  CALL    REGSV
        LD      BC,5
        LDIR
        CALL    REGLD
        RET



Routine:        Err
Function:       Display error message
Called by:      Error, Print
Calls:          None
Entry:          A holds error code
Exit:           None
Preserved:      B,C,D,E,H,L,IX,IY


ERR     PUSH    AF
        AND     15
        OUT     (8),A
        LD      A,14
        OUT     (13),A
        LD      A,33
        OUT     (144),A
        POP     AF
        SRL     A
        SRL     A
        SRL     A
        SRL     A
        OR      A
        RET     Z
; The error code is a single byte. The lower 4 bits are displayed at the right
of the display; if the higher 4 bits are zero the digit is not displayed.
        OUT     (9),A
        LD      A,49
        OUT     (144),A
        RET
```

```
Routine:        Error
Function:       Displays error message and waits for a key to be pressed
Called by:      Chk1, Chk3, Fpint, Measno, Recip, , Vischk
Calls:          Err, Wait
Entry:          A holds error code
Exit:           A holds key code
Preserved:      IX,IY,E


ERROR   CALL    ERR
; Call subroutine Err to display the error, then Wait to pause until a key is
pressed.
        CALL    WAIT
        RET



Routine:        Fpint
Function:       Converts a floating point number to a two byte integer. The FP
number must be positive and <65536.
Called by:      Measno, Visc
Calls:          Error
Entry:          IX points to FP number
Exit:           HL holds the integer part
Preserved:      None


FPINT   LD      B,(IX+4)
        LD      D,(IX+3)
        LD      E,(IX+2)
        LD      H,(IX+1)
        LD      L,(IX)
        LD      A,B
; Read the floating point number into the Z80 registers.
        CP      17
        JR      C,FPINT1
        CP      128
        JR      C,FPINT4
; If the floating point number is too big then jump to fpint4 and signal error
code 6; otherwise if it is zero, set HL to zero and return.
        LD      L,0
        LD      H,L
        RET
FPINT1  BIT     7,D
        JR      NZ,FPINT5
; If negative, signal error 7. The rest of the routine is just shifting until
the exponent becomes zero.
        INC     B
        XOR     A
        LD      C,A
        SET     7,D
        JR      FPINT3
FPINT2  RL      L
        RL      H
        RL      E
        EL      D
        RL      C
        RL      A
FPINT3  DJNZ    FPINT2
; Load the integer part of the floating point number into HL, and return.
        LD      L,C
        LD      H,A
        RET
FPINT4  LD      A,6
```

```
        CALL    ERROR
        RET
FPINT5  LD      A,7
        CALL    ERROR
        RET
```

Routine: Init
Function: Initialises certain variables
Called by: Called automatically straight after a System Reset.
Calls: Follows into Main
Entry: None
Exit: None
Preserved: None

```
; The alarm period is set to its default value of 1 hour, and the time is set to
midnight.
INIT    LD      A,255
        OUT     (142),A
        OUT     (143),A
; Stop the motor.
        IN      A,(6)
        OUT     (CLK50),A
        LD      A,16
        LD      (CLKDOT),A
;Set Clkdot for the start of a second.
        XOR     A
        LD      (CLKFLG),A
        LD      (LMMIN),A
        LD      (LMMIN+1),A
        ÖD      (LMHRS+1),A
        LD      (CLKSEC),A
        LD      (CLKSEC+1),A
        LD      (CLKMIN),A
        LD      (CLKMIN+1),A
        LD      (CLKHRS),A
        LD      (CLKHRS+1),A
        INC     A
        LD      (CLKFLG),A
; The clock is started in format B.
        IN      A,(6)
        LD      (CLK50),A
; Reset the 50 Hz counter.
MAIN
```

Routine: Input
Function: Get a number from the user and convert it into Floating point format
Called by: Almvis, Calib, Measno
Calls: Keyb, Mult, Regld, Regsv
Entry: HL points to variable's location
Exit: Input variable stored at HL
Preserved: All

```
INPUT   CALL    REGSV
        PUSH    HL
INP0    LD      B,5
        LD      A,255
        LD      HL,INPBUF
```

```
; Use storage space at Inpbuf as temporary space.
INP1    LD      (HL),A
        INC     HL
        DJNZ    INP1
; Clear the storage space.
        LD      (HL),0
        LD      A,2
        LD      (INPDGT),A
; Inpdgt holds the number of digits entered +2.
        LD      IX,INPBUF
        LD      A,(IX+5)
        JR      INP7
INP2    CALL    KEYB
        CP      255
        JR      Z,INP2
; Detect a keypress and see if it is a decimal point: if not then jump to Inp4.
        CP      17
        JR      NZ,INP4
        LD      HL,INPDGT
        BIT     7,(HL)
        JR      NZ,INP2
; If 6 digits have already been entered then allow no more!
        SET     7,(HL)
        LD      A,(HL)
        CP      130
        JR      NZ,INP3
        INC     (HL)
INP3    SET     4,(IX+5)
; Enter a decimal point.
        JR      INP2
INP4    CP      21
        JR      Z,INPENT
; 'Enter' causes a jump to the conversion part of the subroutine, 'Clr' causes
the input buffer to be cleared.
        CP      19
        JR      Z,INP0
        CP      10
        JR      P,INP2
; Do not allow a non-numeric keypress.
        OR      A
        JR      NZ,INP5
; A zero entered requires special treatment depending on whether or not it is
the first digit to be entered, in which case it is ignored.
        LD      D,A
        LD      A,(INPDGT)
        CP      2
        JR      Z,INP2
        LD      A,D
INP5    LD      HL,INPDGT
        BIT     3,(HL)
; If 6 digits have already been entered then allow no more! Otherwise increment
the number of digits flag.
        JR      NZ,INP2
        INC     (HL)
        LD      HL,INPBUF+5
        LD      D,A
        LD      A,(INPDGT)
        CP      3
        JR      NZ,INP6
        LD      (HL),255
INP6    LD      A,D
        LD      HL,INPBUF
        LD      DE,INPBUF
        LD      BC,5
```

```
        INC     HL
        LDIR
; Shift the digits in the display buffer left to make room for the typed digit.
        LD      (IX+5),A
        LD      HL,INPDGT
        BIT     7,(HL)
        JR      NZ,INP8
INP7    OR      16
INP8    LD      C,8
        LD      B,6
        LD      HL,INPBUF+5
INP9    OUT     (C),A
        RL      A
        RL      D
        INC     C
        DEC     HL
        LD      A,(HL)
        DJNZ    INP9
; Display all the digits in the display buffer.
        LD      A,D
        CPL
        OUT     (144),A
; Set the digit blanking register according to digits entered. Note that
initially the input buffer contains 255s; these are rotated into the D register
in the above loop, which after inversion gives the correct value to be sent to
the digit blanking register.
        JP      INP2
INPENT  LD      HL,0
; Conversion of data in the input buffer to floating point format.
        LD      D,H
        LD      E,H
        LD      IX,INPBUF
        LD      B,6
        LD      C,0
INP10   PUSH    BC
        LD      A,(IX)
        INC     IX
        CP      255
        JR      NZ,INP11
        XOR     A
; Enter zeros for 255s in the input buffer.
INP11   BIT     4,A
        JR      Z,INP12
; Detect the decimal point when it occurs and save its location in the C
register for later use.
        POP     BC
        LD      C,B
        DEC     C
        PUSH    BC
        AND     15
; The following is the actual decimal to binary conversion for 1 digit; it is
repeated 6 times for the entire input buffer.
INP12   CALL    INPSHF
        LD      (INPHI),DE
        LD      (INPLO),HL
        CALL    INPSHF
        CALL    INPSHF
        LD      BC,(INPLO)
        ADD     HL,BC
        EX      DE,HL
        LD      BC,(INPHI)
        ADC     HL,BC
        EX      DE,HL
        LD      C,A
```

```
        LD      B,0
        ADD     HL,BC
        JR      NC,INP13
        INC     E
INP13   POP     BC
        DJNZ    INP10
; Repeat the conversion 6 times: the counter is held in the B register.
        LD      B,32
        XOR     A
        OR      L
        OR      H
        OR      E
        JR      NZ,INP15
; If zero was entered, the following section inserts a floating point zero at
the desired input variable location, then returns. Floating point zero is
defined as 0,0,0,0,128. For non-zero numbers jump to Inp15.
        POP     HL
        LD      B,4
INP14   LD      (HL),0
        INC     HL
        DJNZ    INP14
        LD      (HL),128
        CALL    REGLD
        RET
; Normalise the number.
INP15   SLA     A
        RL      H
        RL      E
        RL      D
        DEC     B
        BIT     7,D
        LD      (INPBUF),HL
        LD      (INPBUF+2),DE
        LD      A,B
        LD      (INPBUF+4),A
        LD      B,C
        LD      A,B
        OR      A
        JR      Z,INP17
; If  an integer was entered then no multiplications are necessary (jump to
Inp17); otherwise multiply by a tenth, C number of times, to take account of the
position of the decimal point in the input buffer.
INP16   PUSH    BC
        LD      BC,INPBUF
        LD      DE,TENTH
        LD      HL,INPBUF
        CALL    MULT
        POP     BC
        DJNZ    INP16
INP17   POP     HL
        LD      IX,INPBUF
        LD      B,5
INP18   LD      A,(IX)
        LD      (HL),A
        INC     HL
        INC     IX
        DJNZ    INP18
; Transfer the result to the desired variable location specified by HL and
return.
        CALL    REGLD
        RET
INPSHF  ADD     HL,HL
        EX      DE,HL
        ADC     HL,HL
```

```
        EX      DE,HL
; This subroutine shifts the DE and HL register pairs by 1 bit left.
        RET
```

```
Routine:        Intfp
Function:       Converts a two byte positive integer to floating point format.
Called by:      Pc, Range.
Calls:          None
Entry:          HL holds the integer, IX points to the Floating point variable.
Exit:           FP stored at IX
Preserved:      None
```

```
INTFP   XOR     A
        OR      H
        OR      L
        JR      NZ,INTFP1
; If the number to be converted is zero, enter the floating point zero, defined
as 0,0,0,0,128, and return.
        LD      B,128
        JR      INTFP3
INTFP1  LD      B,16
; Rotate the integer, decreasing the exponent, until the leftmost bit is a 1.
INTFP2  BIT     7,H
        JR      NZ,INTFP3
        ADD     HL,HL
        DEC     B
        JR      INTFP2
INTFP3  RES     7,H
        LD      (IX+4),B
        LD      (IX+3),H
        LD      (IX+2),L
        XOR     A
        LD      (IX+1),A
        LD      (IX+0),A
; Store the floating point number at the specified location.
        RET
```

```
Routine:        Keyb
Function:       Detect a keypress and update the clock
Called by:      Almset, Clkset, Input, Main, Pb3, Pc, Wait
Calls:          Clock
Entry:          None
Exit:           A holds keycode
Preserved:      IX,IY,E
```

```
KEYB    CALL    CLOCK
; Update the clock each time Keyb is called.
        LD      C,8
        LD      D,0
        LD      L,255
; Scan the keyboard matrix.
KEYB0   IN      A,(C)
        OR      224
        CP      255
        JR      Z,KEYB3
; If no key in row C is pressed, then go to the next row, otherwise find out
which key was pressed, by rotating until a '0' is found.
        LD      B,5
KEYB1   RR      A
```

```
        JR      C,KEYB2
        LD      H,A
        LD      A,255
        CP      L
; Check that this is the first key detected. If more than one key is pressed
then jump to Keyb7, i.e disregard them all.
        JR      NZ,KEYB7
        LD      L,D
        LD      A,H
KEYB2   INC     D
        DJNZ    KEYB1
KEYB3   LD      A,5
        ADD     D
        LD      D,A
        LD      A,12
        INC     C
        CP      C
        JR      NZ,KEYB0
; Return to Keyb0 to scan a new row if they have not all been scanned already.
        LD      A,(KEYLST)
        CP      L
        JR      NZ,KEYB4
; Check that the currently pressed key is not the same as the previous key. If
it is then return 255, i.e no key pressed. This prevents auto-repeating.
        LD      A,255
        RET
KEYB4   LD      A,L
        LD      (KEYLST),A
; Store the new keynumber and wait for 3/50 of a second. This effectively
debounces the key.
        IN      A,(6)
        LD      D,A
KEYB5   IN      A,(6)
        SUB     D
        JP      P,KEYB6
        NEG
KEYB6   CP      3
        JR      C,KEYB5
        LD      A,255
        CP      L
        RET     Z
; Return if no keys were pressed, otherwise add the keynumber to the start
address of the list of keycodes and fetch the keycode.
        LD      BC,KEYS
        LD      H,0
        ADD     HL,BC
        LD      A,(HL)
        CP      16
; Check that the pressed key was not Rst, if it was then jump to Keyb7 and
return 255, i.e no key pressed. A Rst keypress requires special treatment and is
detected by the subroutine Reset.
        JR      Z,KEYB7
        RET
KEYB7   LD      A,255
        RET
```

Routine: Main
Function: Main driver routine
Called by: Preceded directly by Init
Calls: Almset, Almvis, Calib, Clkset, Keyb, Meas, Measno, Pb2, Pb3, Pb4, Pc, Print, Wait
Entry: None

```
Exit:           None
Preserved:      None


MAIN    LD      IX,CLKFLG
        SET     7,(IX)
; Enable the clock display.
        CALL    WAIT
; Wait for a keypress, disable clock display, and see if the key was 'A'. If not
jump to Main6.
        RES     7,(IX)
        CP      10
        JR      NZ,MAIN6
        OUT     (13),A
        LD      A,1
        OUT     (144),A
MAIN1   CALL    KEYB
        CP      19
; If the Clr key is pressed then restart.
        JR      Z,MAIN
        LD      B,1
        CALL    MAIN13
; Calling the Main13 subroutine checks the next character entered to see if its
a 1, if so then subroutine Clkset is called.
        JR      NZ,MAIN2
        CALL    CLKSET
        JR      MAIN
MAIN2   LD      B,2
        CALL    MAIN13
        JR      NZ,MAIN3
; Program A2, set measurement period.
        CALL    ALMSET
        JR      MAIN
MAIN3   LD      B,3
        CALL    MAIN13
        JR      NZ,MAIN4
; Program A3, Calibrate.
        CALL    CALIB
        JR      MAIN
MAIN4   LD      B,4
        CALL    MAIN13
        JR      NZ,MAIN5
; Program A4, Set measurement range.
        CALL    ALMVIS
        JR      MAIN
MAIN5   LD      B,5
        CALL    MAIN13
        JR      NZ,MAIN1
; Program A5, Get number of measurements.
        CALL    MEASNO
        JR      MAIN
MAIN6   CP      11
; Check if key was 'B', if not, jump.
        JR      NZ,MAIN11
        OUT     (13),A
        LD      A,1
        OUT     (144),A
MAIN7   CALL    KEYB
        CP      19
        JR      Z,MAIN
        LD      B,1
        CALL    MAIN13
        JR      NZ,MAIN8
; Program B1, one-off viscosity measurement.
```

```
        LD      HL,VISCTY
        CALL    MEAS
        CALL    PRINT
        CALL    WAIT
        JP      MAIN
MAIN8   LD      B,2
        CALL    MAIN13
        JR      NZ,MAIN9
; Program B2, continuous viscosity measurement.
        CALL    PB2
        JP      MAIN
MAIN9   LD      B,3
        CALL    MAIN13
        JR      NZ,MAIN10
; Program B3, timed viscosity measurements, clock shows real time.
        CALL    PB3
        JP      MAIN
MAIN10  LD      B,4
        CALL    MAIN13
        JR      NZ,MAIN7
; Program B4, timed viscosity measurements, clock shows time elapsed since start
of experiment.
        CALL    PB4
        JP      MAIN
MAIN11  CP      12
; Check if the key pressed was 'C', if not, start Main again.
        JP      NZ,MAIN
        OUT     (13),A
        LD      A,1
        OUT     (144),A
MAIN12  CALL    KEYB
        CP      19
        JP      Z,MAIN
        C       22
        JR      NZ,MAIN12
; Program C, scan through results.
        CALL    PC
        JP      MAIN
MAIN13  CP      B
; Subroutine Main13 checks the entered key against the code held in the B
register and returns if not equal. If equal it waits until the Run key is
pressed.
        RET     NZ
        OUT     (12),A
        LD      A,3
        OUT     (144),A
MAIN14  CALL    KEYB
        CP      19
        JR      NZ,MAIN15
; If Clr key pressed, Pop AF to restore the stack, and jump to the start of
Main.
        POP     AF
        JP      MAIN
MAIN15  CP      22
        RET     Z
        JR      MAIN14
```

```
Routine:        Mark
Function:       Counts "Mark" of the Mark/Space ratio
Called by:      Read
Calls:          None
Entry:          HL points to the location mark will be stored at
```

```
Exit:           Mark stored at HL
Preserved:      None


MARK    IN      A,(3)
        LD      E,A
        IN      A,(4)
        LD      D,A
        IN      A,(5)
        LD      C,A
; Input the counter values through ports 3, 4 and 5.
        LD      B,24
        XOR     A
MARK1   BIT     7,C
        JR      NZ,MARK2
; Normalise by shifting left until the leftermost bit is a 1.
        SLA     E
        RL      D
        RL      C
        DEC     A
        DJNZ    MARK1
MARK2   RES     7,C
; Store the result at HL.
        LD      (HL),0
        INC     HL
        LD      (HL),E
        INC     HL
        LD      (HL),D
        INC     HL
        LD      (HL),C
        INC     HL
        LD      (HL),A
        RET


Routine:        Meas
Function:       Actual viscosity measurement
Called by:      Main, Pb2, Pb3
Calls:          Mult, Regld, Regsv, Visc
Entry:          HL holds address viscosity is to be stored at
Exit:           Viscosity stored at HL; C flag is set on error
Preserved:      All


MEAS    CALL    REGSV
        CALL    VISC
; Raw viscosity measurement.
        RET     C
        LD      B,H
        LD      C,L
        LD      DE,CALVIS
; Multiply by the calibration coefficient.
        CALL    MULT
        CALL    REGLD
        RET


Routine:        Measno
Function:       Get from the user the number of readings required and check that
its not too many
Called by:      Main
Calls:          Error, Fpint, Input
Entry:          None
```

```
Exit:           lognmx holds number of measurements required
Preserved:      None


MEASNO  LD      HL,NUM1
        CALL    INPUT
; Get the number from the user.
        PUSH    HL
        POP     IX
        CALL    FPINT
; Convert from floating point to two byte integer.
        LD      (LOGNMX),HL
        LD      DE,LOGMAX
        OR      A
        SBC     HL,DE
; Subtract maximum number of measurements allowed and return if less, otherwise
indicate error A, and get another value.
        RET     C
        LD      A,10
        CALL    ERROR
        JR      MEASNO



Routine:        Mlt
Function:       Multiplies two 32-bit positive integers
Called by:      Mult, Print
Calls:          None
Entry:          #1 in Mltbuf 0-3, #2 in Mltbuf 4-7
Exit:           Result in Mltbuf 8-15
Preserved:      C,D,E,H,L,IY


MLT     LD      IX,MLTBUF
        XOR     A
        LD      (IX+8),A
        ÖD      (IX+9),A
        ÖD      (IX+10),A
        LD      (IX+11),A
; Clear space for result bytes 0-3.
        LD      B,32
; In the following loop, 32 bit multiplication is acheived by repeatedly
shifting the result/multiplier and adding the multiplicand if a 1 is shifted
out.
MLT1    SLA     (IX+8)
        RL      (IX+9)
        RL      (IX+10)
        RL      (IX+11)
        RL      (IX)
        RL      (IX+1)
        RL      (IX+2)
        RL      (IX+3)
        JR      NC,MLT2
        LD      A,(IX+8)
        ADD     A,(IX+4)
        LD      (IX+8),A
        LD      A,(IX+9)
        ADC     A,(IX+5)
        LD      (IX+9),A
        LD      A,(IX+10)
        ADC     A,(IX+6)
        LD      (IX+10),A
        LD      A,(IX+11)
        ADC     A,(IX+7)
```

```
        LD      (IX+11),A
        JR      NC,MLT2
        INC     (IX)
        JR      NZ,MLT2
        INC     (IX+1)
        JR      NZ,MLT2
        INC     (IX+2)
        JR      NZ,MLT2
        INC     (IX+3)
MLT2    DJNZ    MLT1
; Store bytes 4-7 in Mltbuf 8-15.
        LD      A,(IX)
        LD      (IX+12),A
        LD      A,(IX+1)
        LD      (IX+13),A
        LD      A,(IX+2)
        LD      (IX+14),A
        LD      A,(IX+3)
        LD      (IX+15),A
        RET
```

```
Routine:        Mult
Function:       Multiply two floating point numbers
Called by:      Divide, Input, Meas, Print, Read, Visc
Calls:          Mlt, Norm, Regld, Regsv
Entry:          BC points to #1, DE points to #2, HL points to result
Exit:           Result stored at HL
Preserved:      All
```

```
MULT    CALL    REGSV
        PUSH    HL
        PUSH    BC
        LD      IX,MLTBUF
        CALL    MULT7
; Move number 2 to mltbuf.
        LD      A,(DE)
        LD      (MULTE),A
; Exponents are held in Multe.
        POP     DE
        CALL    MULT7
; Move number 1 to Mltbuf+4.
        LD      A,(DE)
        LD      B,A
        LD      A,(MULTE)
        ADD     A,B
; Add the exponents.
        LD      (MULTE),A
        LD      A,(MLTBUF+3)
        BIT     7,A
        JR      NZ,MULT1
; Check the sign of number2: the B register is set to 0 if positive, 255 if
negative.
        LD      B,0
        SET     7,A
        LD      (MLTBUF+3),A
        JR      MULT2
MULT1   LD      B,255
MULT2   LD      A,(MLTBUF+7)
        BIT     7,A
        JR      NZ,MULT3
        SET     7,A
```

```
        LD      (MLTBUF+7),A
; Check the sign of number 1, if it is positive then the result sign is that of
number two, otherwise invert the sign. The sign is held in Mltsgn.
        LD      A,B
        JR      MULT4
MULT3   LD      A,B
        CPL
MULT4   LD      (MLTSGN),A
; 32 bit multiplication is performed by Mlt, and the result normalised by
calling subroutine Norm.
        CALL    MULT
        CALL    NORM
        LD      A,(MLTSGN)
        OR      A
        JR      NZ,MULT5
        RES     7,(IX+15)
; Insert the sign bit into the result.
MULT5   POP     HL
        LD      IX,MLTBUF+12
        LD      B,4
; Move the result to the location pointed to by HL.
MULT6   LD      A,(IX)
        LD      (HL),A
        INC     HL
        INC     IX
        DJNZ    MULT6
        LD      A,(MULTE)
; Store exponent byte also.
        LD      (HL),A
        CALL    REGLD
        RET
MULT7   LD      B,4
; Subroutine Mult7 is used to transfer the two numbers into space at Mltbuf.
MULT8   LD      A,(DE)
        LD      (IX),A
        INC     DE
        INC     IX
        DJNZ    MULT8
        LD      A,(DE)
        RET
```

Routine: Norm
Function: Normalise a floating point number
Called by: Add, Mult, Recip
Calls: None
Entry: 64-bit number is from IX+8 to IX+15
Exit: 64-bit number is from IX+8 to IX+15, Multe is adjusted as necessary
Preserved: C,D,E,H,L,IX,IY

```
NORM    LD      B,32
NORM1   LD      A,(IX+15)
; Normalise by shifting left and decrementing the exponent until a 1 appears in
the Most Significant Place.
        BIT     7,A
        RET     NZ
        SLA     (IX+8)
        RL      (IX+9)
        RL      (IX+10)
        RL      (IX+11)
        RL      (IX+12)
```

```
        RL      (IX+13)
        RL      (IX+14)
        RL      (IX+15)
        LD      A,(MULTE)
        DEC     A
        LD      (MULTE),A
        DJNZ    NORM1
        RET
```

```
Routine:        Pb2
Function:       Program B2: Continuous viscosity measurement
Called by:      Main
Calls:          Meas, Print, Vischk, Wait
Entry:          None
Exit:           As from Wait
Preserved:      None
```

```
PB2     LD      HL,VISCTY
; Measure, Check it falls within the set range, print, and wait for a keypress.
        CALL    MEAS
        CALL    PRINT
        CALL    VISCHK
        JR      Z, PB2
        LD      HL,VISCTY
        CALL    PRINT
        CALL    WAIT
        RET
```

```
Routine:        Pb3
Function:       Program B3: Timed measurements
Called by:      Main, Pb4
Calls:          Alarm, Alm2, Equals, Keyb, Meas, Print, Vischk.
Entry:          None
Exit:           None
Preserved:      None
```

```
PB3     CALL    ALM2
; Set the next alarm time.
        LD      HL,LOG
        LD      (LOGCUR),HL
; Initialise the result pointers to the start of the result area, and set
measurement number zero.
        LD      HL,0
        LD      (LOGN),HL
        LD      IX,CLKFLG
        SET     7,IX
; Display the clock, and jump to the first measurement.
        JR      PB3D
PB3A    CALL    KEYB
; Ckeck the keyboard; if ´B' pressed, display the time, if 'A' pressed, display
the last viscosity measured.
        LD      IX,CLKFLG
        CP      11
        JR      NZ,PB3B
        SET     7,(IX)
        JR      PB3C
PB3B    CP      10
        JR      NZ,PB3C
        RES     7,(IX)
```

```
        LD      HL,VISCTY
; Display last viscosity.
        CALL    PRINT
PB3C    CALL    ALARM
; Check to see if alarm period has expired yet. If so, take a reading.
        JR      NZ,PB3A
PB3D    LD      HL,VISCTY
        CALL    MEAS
; Measure and print the viscosity, and store it at the current result location.
        CALL    PRINT
        LD      DE,(LOGCUR)
        CALL    EQUALS
        LD      HL,(LOGCUR)
; Increment the result pointer.
        INC     HL
        INC     HL
        INC     HL
        INC     HL
        INC     HL
        LD      (LOGCUR),HL
        LD      BC,(LOGNMX)
        LD      DE,(LOGN)
        INC     DE
        LD      (LOGN),DE
; Increment the result number, and check that it does not equal the number of
results requested by the user.
        LD      A,C
        CP      E
        JR      NZ,PB3E
        LD      A,B
        CP      D
; Return if all measurements are done.
        RET     Z
PB3E    CALL    VISCHK
; Check that the viscosity is within the user defined range, and return if not.
        JR      Z,PB3A
        RET
```

Routine: Pb4
Function: Program B4: As Pb3 except program B4 displays time elapsed since start of experiment, not real-time
Called by: Main
Calls: Pb3
Entry: None
Exit: None
Preserved: None

```
PB4     LD      IX,CLKSEC
        XOR     A
        LD      B,6
; Set all time bytes to zero and reset the 50 Hz counter, then jump to Pb3.
PB4A    LD      (IX),A
        INC     IX
        DJNZ    PB4A
        IN      A,(6)
        LD      (CLK50),A
        JP      PB3
```

Routine: Pc
Function: Program C: Scan through the measurements

```
Called by:      Main
Calls:          Intfp, Keyb, print
Entry:          None
Exit:           None
Preserved:      None


PC      LD      HL,0
; Initialise the result pointer and the result number.
        LD      (LOGN),HL
        LD      HL,LOG
        LD      (LOGCUR),HL
        XOR     A
        LD      (LOGFLG),A
; Logflg holds the display format: if 0, the measurement number is displayed, if
1, the measured viscosity is displayed.
PCA     LD      A,(LOGFLG)
        OR      A
        JR      NZ,PCB
        LD      HL,(LOGN)
; Convert measurement number to floating point format, and display.
        LD      IX,NUM1
        CALL    INTFP
        LD      HL,NUM1
        CALL    PRINT
        JR      PCC
PCB     LD      HL,(LOGCUR)
; Print measured viscosity.
        CALL    PRINT
PCC     CALL    KEYB
        CP      19
; Return from Program C if Clr key is pressed.
        RET     Z
        CP      10
        JR      NZ,PCD
; If 'A' pressed, enter display format 0.
        XOR     A
        LD      (LOGFLG),A
        JR      PCA
PCD     CP      11
        JR      NZ,PCE
; If 'B' pressed, enter display format 1.
        LD      (LOGFLG),A
        JR      PCA
PCE     CP      20
; If '+' pressed, then increment the result pointer to the next result, and
increment the result number.
        JR      NZ,PCF
        LD      HL,(LOGCUR)
        INC     HL
        INC     HL
        INC     HL
        INC     HL
        INC     HL
        LD      (LOGCUR),HL
        LD      HL,(LOGN)
        INC     HL
        LD      (LOGN),HL
        LD      DE,(LOGNMX)
; Check that the result number is not more than the number of results there are,
and restart Pc if it is.
        LD      A,L
        CP      E
        JR      NZ,PCA
```

```
        LD      A,H
        CP      D
        JR      NZ,PCA
        JR      PC
PCF     CP      18
; If '-' key pressed, then decrement the result pointer to the next result, and
decrement the result number.
        JR      NZ,PCC
        LD      HL,(LOGCUR)
        DEC     HL
        DEC     HL
        DEC     HL
        DEC     HL
        DEC     HL
        LD      (LOGCUR),HL
        LD      HL,(LOGN)
        DEC     HL
        LD      (LOGN),HL
        LD      A,255
        CP      H
; Check that the result number is not now negative, and restart Pc if it is.
        JR      NZ,PCA
        JP      Z,PC
```

Routine: Print
Function: Displays a floating point number
Called by: Calib, Main, Pb2, Pb3, Pc
Calls: Err, Mlt, Mult, Regld, Regsv
Entry: HL points to the number
Exit: Error 1: Print overflow, Error 2: Print negative attempted
Preserved: All

```
PRINT   CALL    REGSV
        XOR     A
        LD      (PRIOVR),A
        LD      (PRIEXP),A
; Priovr is a flag indicating print overflow, Priexp holds the decimal exponent
of the number, Pribuf is a 7 byte temporary storage space.
        LD      BC,5
        LD      DE,PRIBUF
        LDIR
; Transfer the number into Pribuf.
        LD      IY,PRIEXP
        LD      A,(PRIBUF+3)
        RL      A
; Check the number is positive, if not jump to Prin21, to signal error 2.
        JP      C,PRIN21
        LD      A,(PRIBUF+4)
        CP      128
; Investigate size of the number: if less than 1, call Prin15; if greater than
1, call Prin14. These routines reduce the magnitude of the binary exponent.
        CALL    C,PRIN14
        CALL    NC,PRIN15
        LD      IX,PRIBUF
        SET     7,(IX+3)
        OR      A
        JR      Z,PRIN2
        LD      B,A
; Multiply by two by shifting right, until the binary exponent becomes zero.
This completes the binary to decimal conversion.
PRIN1   SRL     (IX+3)
```

```
        RR      (IX+2)
        RR      (IX+1)
        RR      (IX)
        DJNZ    PRIN1
PRIN2   LD      IX,MLTBUF
        LD      HL,PRIBUF
        LD      BC,4
        LD      DE,MLTBUF
        LDIR
; Move the number to mltbuf.
        LD      B,7
        LD      IY,PRIBUF
; The next part converts the binary fraction to decimal by repeatedly
multiplying by ten and taking the integer part, until seven decimal digits have
been built up in Pribuf.
PRIN3   LD      A,10
        LD      (IX+4),A
        XOR     A
        LD      (IX+5),A
        LD      (IX+6),A
        LD      (IX+7),A
; Put 10 in Mltbuf 4-7
        PUSH    BC
        CALL    MLT
        POP     BC
        LD      A,(IX+12)
        LD      (IY),A
; Store integer part in Pribuf and move the result back into Mltbuf 0-3.
        INC     IY
        LD      HL,(MLTBUF+8)
        LD      (MLTBUF),HL
        LD      HL,(MLTBUF+10)
        LD      (MLTBUF+2),HL
        DJNZ    PRIN3
        LD      A,(PRIEXP)
        DEC     A
        CP      127
; Investigate the decimal exponent; if the number is less than 1, call Prin9, if
greater than 1, continue at Prin4.
        JR      C,PRIN4
        CALL    PRIN9
        LD      A,(PRIOVR)
        OR      A
        JR      Z,PRIN5
; If the number is less than 0.00001, then just print '0.' and return, otherwise
jump to prin5.
        LD      A,17
        OUT     (8),A
        LD      A,32
        OUT     (144),A
        JR      PRINZ
PRIN4   CP      6
; If the decimal exponent is too big to display then jump to Prin17 to indicate
error 1.
        JP      NC,PRIN17
        CALL    PRIN18
        LD      A,(PRIOVR)
        OR      A
; Jump to Prin17 in the event of an overflow.
        JP      NZ,PRIN17
PRIN5   LD      A,255
        PUSH    AF
; Next shift out all the leading zeros, using the A register to hold the
information to be sent to the digit blanking register, Out 144.
```

```
PRIN6   LD      A,(PRIBUF+5)
        OR      A
        JR      NZ,PRIN7
        LD      BC,5
        LD      HL,PRIBUF+4
        LD      DE,PRIBUF+5
        LDDR
        POP     AF
        SLA     A
        PUSH    AF
        JR      PRIN6
PRIN7   LD      B,6
        LD      C,13
        LD      HL,PRIBUF
; Output the contents of Pribuf to the displays, and set the digit blanking
register, Out 144, appropriately.
PRIN8   LD      A,(HL)
        OUT     (C),A
        DEC     C
        INC     HL
        DJNZ    PRIN8
        POP     AF
        OUT     (144),A
; Jump to the routine end.
        JR      PRINZ
PRIN9   LD      IX,PRIBUF+5
; First round the number by checking the least significant digit, and if greater
than or equal to 5, incrementing the last digit of the displayed number.
        LD      A,(IX+1)
        CP      5
        JR      C,PRIN11
PRIN10  LD      A,(IX)
        INC     A
        LD      (IX),A
        CP      10
        JR      NZ,PRIN11
        XOR     A
        LD      (IX),A
        DEC     IX
        JR      PRIN10
PRIN11  LD      HL,PRIBUF
        LD      A,(PRIEXP)
        DEC     A
; Create appropriate leading zeros by shifting the number right, until the
exponent becomes zero.
PRIN12  OR      A
        JR      Z,PRIN13
        LD      BC,5
        LD      HL,PRIBUF+4
        LD      DE,PRIBUF+5
        LDDR
        PUSH    AF
        XOR     A
        LD      (PRIBUF),A
        POP     AF
        INC     A
        JR      PRIN12
PRIN13  LD      A,(PRIBUF)
; Insert a decimal point after the leftermost zero, then return.
        OR      16
        LD      (PRIBUF),A
        RET
PRIN14  LD      A,(PRIBUF+4)
; Multiply the number by a tenth until it is less than one, i.e the integer part
```

```
is reduced to zero; for each multiplication increment the decimal exponent.
        LD      B,A
        XOR     A
        SUB     B
        CP      4
        RET     C
        LD      BC,TENTH
        LD      DE,PRIBUF
        LD      HL,PRIBUF
        CALL    MULT
        INC     (IY)
        JR      PRIN14
PRIN15  LD      A,(PRIBUF+4)
; Multiply the number by ten until any futher multiplications would make it
greater than one; for each multiplication, decrement the decimal exponent.
        CP      252
        JR      C,PRIN16
        LD      B,A
        XOR     A
        SUB     B
        RET
PRIN16  LD      BC,TEN
        LD      DE,PRIBUF
        LD      HL,PRIBUF
        CALL    MULT
        DEC     (IY)
        JR      PRIN15
PRIN17  LD      A,1
; Signal print overflow error.
        CALL    ERR
        JR      PRINZ
; Subroutine Prin18 rounds the number, by checking the least significant digit
and if greater than or equal to 5, incrementing the last digit of the displayed
number.
PRIN18  LD      IX,PRIBUF+5
        LD      A,(IX+1)
        CP      5
        JR      C,PRIN20
PRIN19  LD      A,(IX)
        INC     A
        LD      (IX),A
        CP      10
        JR      NZ,PRIN20
; If incrementing the last digit resulted in a number > 10, it must be reset and
the next digit incremented, and so on until no more overflows occur,
        XOR     A
        LD      (IX),A
        DEC     IX
        JR      PRIN19
PRIN20  LD      HL,PRIBUF
        LD      A,(PRIEXP)
        DEC     A
; Calculate position of the decimal point in the print buffer and inser it by Or
16.
        LD      E,A
        LD      D,0
        ADD     HL,DE
        LD      A,16
        OR      (HL)
        LD      (HL),A
        RET
PRIN21  LD      A,2
; Signal print negative error.
        CALL    ERR
```

```
PRINZ   CALL    REGLD
; Terminate Print routine.
        RET
```

```
Routine:        Range
Function:       Finds minimum and maximum motor speeds that are suitable for the
set of measurements
Called by:      Visc
Calls:          Chk1, Chk2, Chk3, Divide, Intfp, Sub
Entry:          None
Exit:           Mtrmin, Mtrmax set; C flag is set on error
Preserved:      None
```

```
RANGE   CALL    CHK1
; Find the maximum motor speed, angle < 170 degrees; return on error.
        RET     C
        CALL    CHK2
; Find minimum motor speed, angle >0.
        CALL    CHK3
; Check max speed is more than min speed, return on error.
        RET     C
        LD      A,(MTRMIN)
        LD      L,0
        LD      H,0
        LD      IX,RNGMAX
; Convert max speed to floating point.
        CALL    INTFP
        LD      A,(MTRMAX)
        LD      L,0
        LD      H,A
        LD      IX,RNGMIN
; Convert min speed to floating point.
        CALL    INTFP
        LD      BC,RNGMAX
        LD      DE,RNGMIN
        LD      HL,RNG
; Calculate range.
        CALL    SUB
        LD      BC,RNG
        LD      DE,TEN
        LD      HL,RNGINC
; Calculate motor speed increment.
        CALL    DIVIDE
        OR      A
        RET
```

```
Routine:        Read
Function:       Reads mechanism: displacement angle and  motor speed.
Called by:      Chk1, Chk2, Visc
Calls:          Add, Clock, Divide, Mark, Mult, Recip, Regld, Regsv, Space
Entry:          DE points to angle, HL to speed
Exit:           Angle, speed stored at DE, HL respectively. NC flag set on error
Preserved:      None
```

```
READ    PUSH    DE
        PUSH    HL
READ1   IN      A,(6)
        LD      (TIMERB),A
```

```
; Initialise variable Timerb: Read6 is used to see that if no change in
mechanism state happens within 2 seconds, the program jumps to Readfl,  and
returns from this routine with the NC flag set. Normal termination is with the C
flag set.
READ2   CALL    READ6
        JP      NC,READFL
        IN      A,(14)
        RL      A
; Read in mechanism state, and repeat until it becomes 0.
        JR      C,READ2
        IN      A,(6)
        LD      (TIMERB),A
READ3   CALL    READ6
        JR      NC,READFL
        IN      A,(14)
        RL      A
; Read in mechanism state and repeat until it becomes 1.
        JR      NC,READ3
        DJNZ    READ1
        IN      A,(7)
        LD      HL,NUM1
        IN      A,(6)
        LD      (TIMERB),A
READ4   CALL    READ6
        JR      NC,READFL
        IN      A,(14)
        RL      A
; Read in mechansim state and repeat until it becomes 0.
        JR      C,READ4
; Now call subroutine Space, to read in the space period.
        CALL    SPACE
        LD      HL,NUM2
        IN      A,(6)
        LD      (TIMERB),A
READ5   CALL    READ6
        JR      NC,READFL
        IN      A,(14)
        RL      A
; Read in mechanism state and repeat until it becomes 1.
        JR      NC,READ5
; Now call subroutine Mark, to read in the mark period.
        CALL    MARK
        LD      BC,NUM1
        LD      DE,NUM2
        LD      HL,NUM3
; Add mark and space periods to get the total rotation period. Next take the
reciprocal to get the motor speed.
        CALL    ADD
        LD      DE,NUM3
        POP     HL
        CALL    RECIP
        LD      BC,NUM2
        LD      DE,NUM3
        LD      HL,NUM4
; Divide Mark by the total period and multiply by 360 degrees, to get the
displacement angle.
        CALL    DIVIDE
        LD      BC,THRE60
        LD      DE,NUM4
        POP     HL
        CALL    MULT
        SCF
        RET
READ6   CALL    REGSV
```

```
; Update clock.
        CALL    CLOCK
        CALL    REGLD
        LD      A,(TIMERB)
        LD      D,A
        IN      A,(6)
        SUB     D
        JP      P,READ7
        NEG
; Check that 2 seconds have not passed.
READ7   CP      100
        RET
READFL  POP     HL
; On error restore stack and return.
        POP     HL
        RET
```

Routine: Recip
Function: Calculates the reciprocal of a floating point number
Called by: Divide, Read
Calls: Error, Norm, Regld, Regsv
Entry: DE points to the number, HL to the result
Exit: Result stored at HL; unchanged on error
Preserved: All

```
RECIP   CALL    REGSV
        PUSH    HL
        LD      HL,MLTBUF
        LD      B,16
        XOR     A
; Clear Mltsgn, used to hold the sign, and clear Mltbuf, used as temporary
space.
        LD      (MLTSGN),A
RCP1    LD      (HL),A
        INC     HL
        DJNZ    RCP1
        LD      L,E
        LD      H,D
        LD      DE,MLTBUF+4
        LD      BC,4
        LDIR
; Transfer the number into Mltbuf+4.
        LD      A,(HL)
        POP     HL
        NEG
        CP      128
; Jump to Rcperr to signal error, on division by floating point zero.
        JP      Z,RCPERR
        PUSH    HL
        INC     A
; Multe holds the exponent part.
        LD      (MULTE),A
        LD      IX,MLTBUF
        BIT     7,(IX+7)
        JR      Z,RCP2
        LD      A,1
; Check the sign and store it in Mltsgn.
        LD      (MLTSGN),A
RCP2    SET     7,(IX+7)
; Mltbuf 0-7 holds the number, Mltbuf 8-11 holds the result, Mltbuf 12-15 holds
the remainder, initially set to 1.
```

```
        SET     7,(IX+15)
; Initially rotate the number right 1 bit, so that the first subtraction does
not automatically result in a negative remainder.
        SRL     (IX+7)
        RR      (IX+6)
        RR      (IX+5)
        RR      (IX+4)
        RR      (IX+3)
        RR      (IX+2)
        RR      (IX+1)
        RR      (IX)
        LD      B,32
; Use the B register as a bit counter: the loop must be run 32 times.
        JR      RCP4
RCP3    SLA     (IX+8)
; Shift the result and remainder left by one bit.
        RL      (IX+9)
        RL      (IX+10)
        RL      (IX+11)
        RL      (IX+12)
        RL      (IX+13)
        RL      (IX+14)
        RL      (IX+15)
RCP4    LD      HL,(MLTBUF+12)
        LD      DE,(MLTBUF+4)
; Here subtract the number from the remainder.
        SBC     HL,DE
        LD      (MLTBUF+12),HL
        LD      HL,(MLTBUF+14)
        LD      DE,(MLTBUF+6)
        SBC     HL,DE
        LD      (MLTBUF+14),HL
        JR      C,RCP5
; If the result is still positive, insert a '1' into the result and loop;
otherwise jump to Rcp5.
        INC     (IX+8)
        JR      RCP6
RCP5    LD      HL,(MLTBUF+12)
; The number is added back to the remainder to make it positive again, and a ´0´
is automatically inserted into the result by doing nothing and just waiting for
a '0' to be shifted in in the next pass through the loop.
        LD      DE,(MLTBUF+4)
        ADD     HL,DE
        LD      (MLTBUF+12),HL
        ÖD      HL,(MLTBUF+14)
        LD      DE,(MLTBUF+6)
        ADC     HL,DE
        LD      (MLTBUF+14),HL
RCP6    DJNZ    RCP3
        LD      IX,MLTBUF-4
; Call subroutine Norm to normalise the result.
        CALL    NORM
        LD      IX,MLTBUF+8
        RES     7,(IX+3)
        LD      A,(MLTSGN)
        OR      A
; Jump if positive, otherwise restore the sign bit to a '1'.
        JR      Z,RCP7
        SET     7,(IX+3)
RCP7    POP     HL
        LD      B,4
; Move the result to the the address pointed to by HL.
RCP8    LD      A,(IX)
        LD      (HL),A
```

```
        INC     HL
        INC     IX
        DJNZ    RCP8
; Store the exponent part too.
        LD      A,(MULTE)
        LD      (HL),A
        JR      RCPZ
RCPERR  XOR     A
        CALL    ERROR
; Restore registers and return.
RCPZ    CALL    REGLD
        RET
```

Routine: Regld
Function: Loads all the registers off the stack
Called by: Almvis, Calib, Divide, Err, Input, Meas, Mult, Print, Read, Recip, Sub, Visc
Calls: None
Entry: None
Exit: All
Preserved: None

```
REGLD   POP     IX
        POP     HL
        POP     DE
        POP     BC
        POP     AF
        POP     IY
        EX      (SP),IX
; Restore subroutine return address.
        RET
```

Routine: Regsv
Function: Saves all the registers to the stack
Called by: Almvis, Calib, Divide, Err, Input, Meas, Mult, Print, Read, Recip, Sub, Visc.
Calls: None
Entry: All
Exit: All
Preserved: All

```
REGSV   EX      (SP),IX
        PUSH    IY
        PUSH    AF
        PUSH    BC
        PUSH    DE
        PUSH    HL
        PUSH    IX
; Restore subroutine return address.
        RET
```

outine: Reset
Function: Restarts the entire system, if RST is pressed for longer than 0.5s
Called by: Clock.
Calls: None
Entry: None
Exit: None

Preserved: C,D,E,H,L,IX,IY

```
RESET   IN      A,(8)
        OR      224
        CP      239
; Check if Rst key is being pressed, return if not.
        RET     NZ
        IN      A,(6)
        ADD     A,25
; Load B with current 50 Hz counter value + 25.
        LD      B,A
RESET1  IN      A,(8)
        OR      224
        CP      239
; Check that the Rst key is still pressed.
        RET     NZ
        IN      A,(6)
        CP      B
; Loop until 0.5 seconds have passed.
        JR      NZ,RESET1
; Jump to 0, i.e a system reset.
        RST     0
```

Routine: Space
Function: Counts "Space" of mark/space ratio
Called by: Read
Calls: None
Entry: HL point to the location Space will be stored at
Exit: Space stored at HL
Preserved: None

```
SPACE   IN      A,(0)
        LD      E,A
        IN      A,(1)
        LD      D,A
        IN      A,(2)
        LD      C,A
; Read ports 0-2 into E,D,C registers.
        LD      B,24
        XOR     A
SPACE1  BIT     7,C
; Test the leftermost bit and jump if its a '1', otherwise normalise by shifting
one bit to the left, and repeating.
        JR      NZ,SPACE2
        SLA     E
        RL      D
        RL      C
        DEC     A
; Reduce the exponent, held in A.
        DJNZ    SPACE1
SPACE2  RES     7,C
Store the result at HL.
        LD      (HL),0
        INC     HL
        LD      (HL),E
        INC     HL
        LD      (HL),D
        INC     HL
        LD      (HL),C
        INC     HL
```

```
        LD      (HL),A
        RET
```

Routine: Sub
Function: Subtracts two floating point numbers
Called by: Chk1, Range, Visc, Vischk.
Calls: Add, Regld, Regsv
Entry: BC points to #1, DE to #2, HL to result #1-#2
Exit: Result stored at HL
Preserved: All

```
SUB     CALL    REGSV
        LD      BC,5
        LD      L,E
        LD      H,D
        LD      DE,PRIBUF
        LDIR
; Transfer the second number into temporary storage space at Pribuf.
        LD      IX,PRIBUF
        RL      (IX+3)
        CCF
; Invert the sign bit of the second number's copy at Pribuf.
        RR      (IX+3)
        CALL    REGLD
        PUSH    DE
        LD      DE,PRIBUF
; Call subroutine add; inverting the second number's sign bit makes this
addition effectively the required subtraction.
        CALL    ADD
        POP     DE
        RET
```

Routine: Visc
Function: Reads the absolute, uncalibrated viscosity, does a least squares best fit on the data
Called by: Calib, meas
Calls: Add, Divide, Equals, Fpint, Mult, Range, Read, Regld, Regsv, Sub
Entry: HL points to address of viscosity variable
Exit: C flag is set on error; viscosity is stored at HL
Preserved: All

```
VISC    CALL    REGSV
        PUSH    HL
; Set the statistical variables initially equal to zero.
        LD      DE,STATXX
        LD      HL,ZERO
        CALL    EQUALS
        LD      DE,STATXY
        CALL    EQUALS
        LD      DE,STATYY
        CALL    EQUALS
        LD      DE,STATX
        CALL    EQUALS
        LD      DE,STATY
        CALL    EQUALS
        CALL    RANGE
; Call subroutine Range to find the range of measurement.
        RET     C
        XOR     A
```

```
        OUT     (143),A
        LD      DE,NUM6
        LD      HL,NUM7
        LD      B,10
; Call read with ten revolutions at maximum speed.
        CALL    READ
        LD      B,10
; Use B as an index variable, to count the 10 readings.
VISC1   PUSH    BC
        LD      IX,RNGMIN
Convert Rngmin to integer and output this motor speed to the DAC.
        CALL    FPINT
        LD      A,L
        OUT     (142),A
        LD      A,H
        OUT     (143),A
        LD      DE,NUM6
        LD      HL,NUM7
        LD      B,10
; Take the reading, with 10 revolutions.
        CALL    READ
; Now compute the various statistical variables that are required for the best
fit. See text for explanation.
        LD      BC,NUM7
        LD      DE,NUM7
        LD      HL,NUM8
        CALL    MULT
        LD      BC,STATXX
        LD      DE,NUM8
        LD      HL,STATXX
        CALL    ADD
        LD      BC,NUM7
        LD      DE,NUM6
        LD      HL,NUM8
        CALL    MULT
        LD      BC,STATXY
        LD      DE,NUM8
        LD      HL,STATXY
        CALL    ADD
        LD      BC,NUM6
        LD      DE,NUM6
        LD      HL,NUM8
        CALL    MULT
        LD      BC,STATYY
        ÖD      DE,NUM8
        LD      HL,STATYY
        CALL    ADD
        LD      BC,NUM7
        LD      DE,STATX
        LD      HL,STATX
        CALL    ADD
        LD      BC,NUM6
        LD      DE,STATY
        LD      HL,STATY
        CALL    ADD
        LD      BC,RNGMIN
        LD      DE,RNGINC
        LD      HL,RNGMIN
; Add the range increment to Rngmin to get the new motor speed.
        CALL    ADD
        POP     BC
; Loop if 10 readings have not yet been taken.
        DJNZ    VISC2
        JR      VISC3
```

```
VISC2   JP      VISC1
VISC3   LD      BC,STATX
; Final calculation of the best fit.
        LD      DE,STATX
        LD      HL,NUM8
        CALL    MULT
        LD      BC,NUM8
        LD      DE,TEN
        CALL    DIVIDE
        LD      BC,STATXX
        LD      DE,NUM8
        LD      HL,STATXX
        CALL    SUB
        LD      BC,STATX
        LD      DE,STATY
        LD      HL,NUM8
        CALL    MULT
        LD      BC,NUM8
        LD      DE,TEN
        CALL    DIVIDE
        LD      BC,STATXY
        LD      DE,NUM8
        LD      HL,STATXY
        CALL    SUB
        LD      BC,STATY
        LD      DE,STATY
        LD      HL,NUM8
        CALL    MULT
        LD      BC,NUM8
        LD      DE,TEN
        CALL    DIVIDE
        LD      BC,STATYY
        LD      DE,NUM8
        LD      HL,STATYY
        CALL    SUB
        LD      BC,STATXY
        LD      DE,STATXX
        POP     HL
; Viscosity now returned in HL.
        CALL    DIVIDE
        LD      A,255
        OUT     (143),A
        CALL    REGLD
        OR      A
        RET
```

Routine: Vischk
Function: Check that the viscosity is within the user defined range
Called by: Pb2, Pb3.
Calls: Error, Sub
Entry: Viscty holds th viscosity value to be checked, Vismin and Vismax hold the preset values
Exit: Z Flag set if viscosity is within the range
Preserved: None

```
VISCHK  LD      BC,VISCTY
        LD      DE,VISMIN
        LD      HL,NUM1
        CALL    SUB
; Subtract Vismin from the viscosity, and signal error 8 if the result is
negative.
```

```
        PUSH    HL
        POP     IX
        BIT     7,(IX+3)
        JR      Z,VISCH1
        LD      A,8
        CALL    ERROR
        INC     A
; Inc A to reset the Z flag, signifying that an error occurred.
        RET
VISCH1  LD      BC,VISMAX
        LD      DE,VISCTY
        LD      HL,NUM1
        CALL    SUB
; Subtract Vismax from the viscosity, and signal error 9 if the result is
negative.
        BIT     7,(IX+3)
        RET     Z
        LD      A,9
        CALL    ERROR
        INC     A
; Inc A to reset the Z flag, signifying that an error occurred.
        RET
```

Routine: Wait
Function: Waits for a keypress
Called by: Error, Main, Pb2
Calls: Keyb
Entry: None
Exit: A holds keycode
Preserved: E, IX, IY

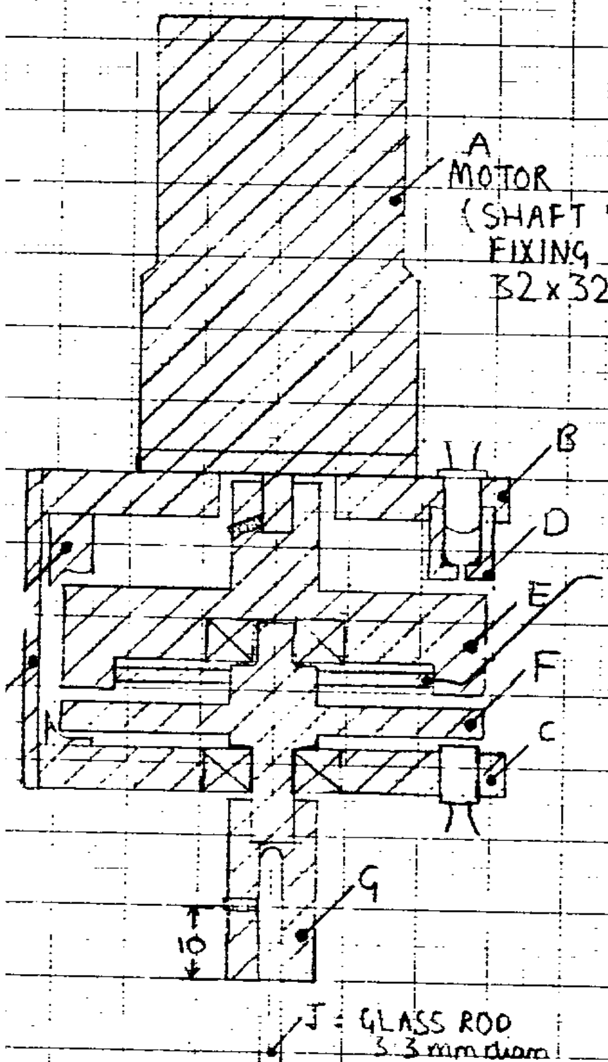
```
WAIT    CALL    KEYB
; Scan keyboard, and loop if no key was pressed.
        CP      255
        JR      Z,WAIT
        RET
```

APPENDIX C: Engineering Drawings

Viscometer Mechanism

29/10/92.

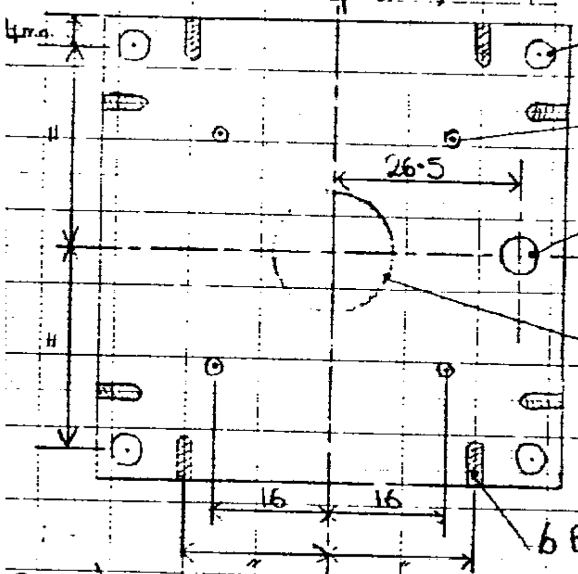
Hans dummers, Tel x 7290

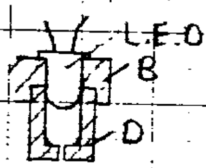


Parts B–H, K made from Dural

A: MOTOR, SHAFT 8mm x 4mm dia, FIXING CENTRES 32x32 square with 2·7 diam holes

B: 6 mm thick plate, 64 x 64 mm

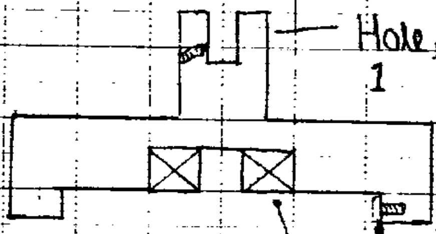




C: Similar to part B, except larger centre hole, to take bearing (push fit) 5mm hole to take photodiode, no recess on one side.
Machine B & C together (NB No motor-fixing holes on part C.)

U: LED cover, fits into recess in B; drill through almost all the way 5mm ID, with small 1mm hole at one end.

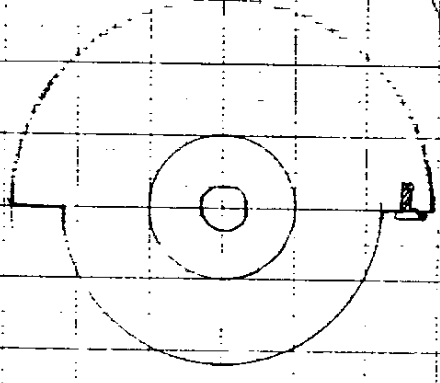
E:



Hole, 4 mm ID, slide fit onto motor shaft, clamped by 1 6 BA grub screws slightly angled so that plate B does not prevent access.

Bearing, push fit.

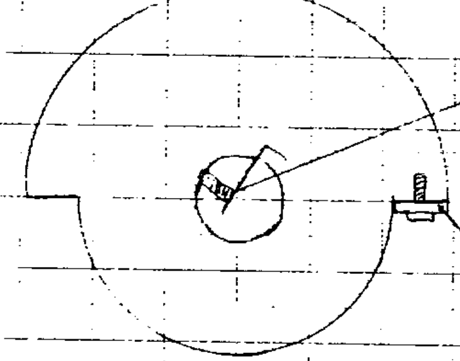
Form recess to take the spiral spring



Cut away 180° of the disk.

Small screw to hold end of spiral spring I

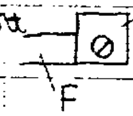
F:



180° outer edge cut away

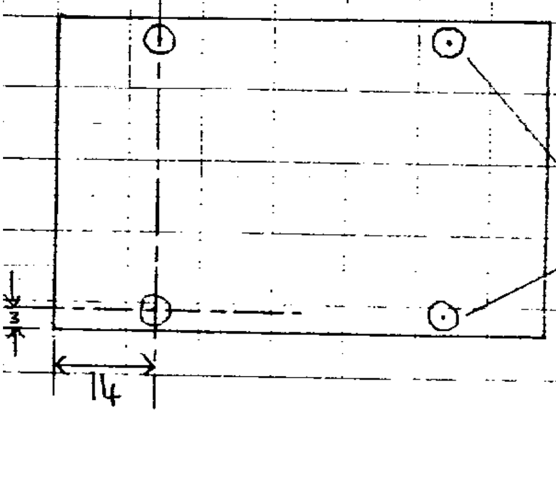
Grub screw to hold other end of spiral spring I; end inserted through cut.

Stop tag: 7 x 7 x 1·5 thick, screwed to F such that angular displacement of E to F is always ≤ half a turn;



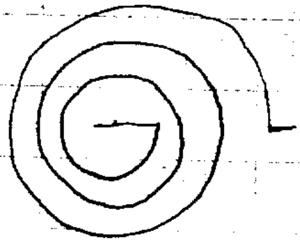
G: slide fit and pin to part F; drilled to accept glass rod, max diam of rod: 3·2mm: clamp rod by 3 6 BA grub screws, tap at 120°

H: 4 off, 1·5 mm thick sheet side-plate, 6 BA clear for screws that mount the side plates to parts B & C. Edges filed to meet adjoining plates;



Note: Pitch of 4 holes to match assembled parts B & C

I: Spiral Spring; made from 2mm strip, very weak



Outside screws to E,
inside end screws to F.

J: Glass rods to be inserted will have a max diam of 3·2mm, so ID of "G" is 3·3 mm.

K: 4 off: Corner pillars from Ø6mm or ¼" rod, length 32mm, 4BA tap in each end, for 4BA x ½ inch screws.

APPENDIX D: Numerical Results

APPENDIX D1: 4.1.3: Angular disulacement and motor speed
(Results are shown to all available decimal places regardless of significance)

300 cst fluid:

Speed: (arbitrary units)	Angle: (degreees)
13.1746	55.6003
12.8795	55.2785
12.7406	59.9503
12.2665	51.8335
11.8283	49.2428
11.339	43.9706
10.9258	43.1309
10.4696	31.6009
9.99599	31.3484
9.49852	20.7161
9.00727	18.3122
8.57492	12.236
8.02222	11.6689

1000 cst fluid:

Speed: (arbitrary units)	Angle: (degreees)
13.152	100.013
12.6899	97.6744
12.2809	103.867
11.8374	90.7453
11.3222	79.9965
10.8544	76.3376
10.397	68.5061
9.89494	65.2955
9.44366	59.6897
8.48747	43.8821
8.03224	33.3045
7.57751	32.9003
7.09929	21.3906
6.63531	15.3734
6.19455	7.11046
5.74406	3.50744

10000 cst fluid:

Speed: (arbitrary units)	Angle: (degreees)
1.53129	18.4551
1.83076	54.8573
2.02239	60.3984
2.22744	75.8374
2.41841	91.6752
2.6425	103.347
2.83324	121.508
3.05397	131.989
3.26053	152.273
3.48993	177.921
3.69334	176.249

APPENDIX D2: 4.2 Dependence of Viscosity on Temperature.

1000 cst DC200 lubricant

Temperature /C	Feraanti-Shirley /cst	Project viscometer /cst
24.4	1000 (calibration)	1000 (calibration)
30.0	914.73	
37.5	818.669	
45.0	761.165	
52.5	660.721	
60.0	612.53	
67.5	521.687	
75.0	492.087	
82.5	411.169	
90.0	389.76	
97.5	411.26	
105.0	382.21	
112.5	321.885	
30.3		870.148
36.1		937.834
50.3		566.022
56.4		332.951
63.0		355.351
71.8		299.268
76.9		275.401
82.5		292.643

92.9		227.168
100.5		158.766

10000 cst DC200 lubricant

Temperature /C	Feraanti-Shirley /cst	Project viscometer /cst
22.5	10000 (calibration)	10000 (calibration)
25.0	9373.596	
30.0	7839.9	
35.0	6112.75	
40.0	5091.6	
45.0	3722.176	
50.0	3084.0	
55.0	2489.8	
60.0	2113.62	
65.0	1740.338	
70.0	1587.7	
75.0	1275.66	
80.0	1149.98	
85.0	922.86	
90.0	841.705	
95.0	712.091	
100.0	636.75	
29.5		9399.01
33.8		8624.73
41.3		6831.83
55.4		1665.88
61.1		1222.41

APPENDIX D3: 4.3 Accuracy and long term stability

Carried out at room temperature on 1000 cst DC200 lubricant. 128 readings, every half hour for 64 hours.

Viscosity readings:

949.558 1001.91 946.369 958.979 877.139 1002.74 974.926 962.344
947.033 1001.93 978.834 976.368 1001.61 869.818 971.86 906.964
949.041 907.625 995.89 996.089 1007.34 934.708 960.586 977.113
977.58 979.538 971.364 965.091 976.265 988.612 985.19 954.478
943.996 967.377 898.123 959.248 960.554 950.565 935.06 930.973
947.318 951.726 977.987 958.262 941.756 921.744 916.805 937.076
1018.84 940.871 846.857 905.924 939.983 916.391 941.499 976.613
971.107 930.231 941.97 965.632 989.284 999.076 991.629 945.657
957.833 912.483 969.333 981.724 925.521 969.501 923.887 962.41

964.472	962.098	878.744	881.49	1008.05	953.882	912.241	909.709
915.393	989.029	1000.47	901.13	969.148	842.929	934.306	863.771
930.226	995.574	998.382	889.828	966.483	973.489	991.375	900.589
1013.52	896.325	913.582	927.039	999.756	955.862	971.057	940.237
1044.38	955.391	910.862	931.014	933.844	919.38	911.379	880.704
962.224	922.067	901.848	923.138	916.182	930.647	942.24	950.281
929.404	927.812	946.464	879.78	945.831	925.414	917.934	933.968